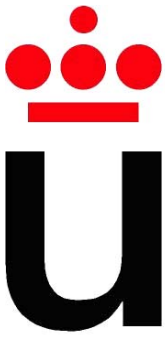

Universidad
Rey Juan Carlos

INGENIERÍA INFORMÁTICA

Curso Académico 2008/2009

Proyecto Fin de Máster

SISTEMA DE ANIMACIÓN INTERACTIVA DE LA RECURSIVIDAD SREC 1.1

Autor: Antonio Pérez Carrasco

Tutor: Ángel Velázquez Iturbide

Resumen

Las visualizaciones de algoritmos se han empleado en las últimas décadas ampliamente en el campo de la docencia. Sin embargo, existen algunos factores que han frenado el desarrollo de esta técnica docente, como son la falta de garantía sobre la eficacia educativa que pueden tener las visualizaciones y el elevado coste de desarrollo por cada una de las visualizaciones que se desea generar

Es por ello que se ha decidido implementar un programa software que visualiza la recursividad sin esfuerzo. Este programa, cuyo nombre es SRec, admite como entrada un archivo con algoritmos escritos en lenguaje Java que es compilado y ejecutado por el usuario aportando los argumentos que estime oportuno, y que posteriormente es mostrado mediante la ventana del programa en su fase de ejecución.

SRec contiene cuatro vistas (tres dinámicas y una estática) que permiten mostrar diferentes aspectos de la recursividad. Una de las vistas ofrece el árbol de activación, permitiendo visualizar todas las subllamadas realizadas. Además, permite el manejo de visualizaciones grandes gracias su funcionalidad de zoom y al visor de navegación que incorpora.

La segunda vista es la pila de control, que mantiene las subllamadas pendientes en cada momento. La tercera vista ofrece, en modo de texto, la sucesión de llamadas y retornos efectuados hasta el momento. Por último, SRec cuenta con una vista estática que permite ver el código del algoritmo que se está ejecutando.

El programa permite la navegación sobre la visualización de manera manual o bien mediante la activación de animaciones. Éstas permiten ver el transcurso de la ejecución del algoritmo recursivo paso a paso.

SRec también aporta una serie de facilidades educativas, como por ejemplo la posibilidad de exportar e importar visualizaciones, lo que permite recuperar visualizaciones sin necesidad de regenerarlas cada vez (facilitando la reutilización de visualizaciones y el intercambio de las mismas entre profesores y alumnos). Además, SRec permite configurar las visualizaciones de múltiples formas, tanto seleccionando qué cantidad de información se muestra como en qué condiciones y con qué formato.

La aplicación da soporte para múltiples idiomas a través de su interfaz, si bien inicialmente el programa estará disponible únicamente en español e inglés.

Sobre SRec se han realizado dos evaluaciones de usabilidad entre alumnos, que complementan a la prueba de uso realizada por el profesor durante las clases magistrales. Los resultados han ayudado a mejorar sensiblemente el programa.

Índice

1. Introducción

En la sociedad actual, la tecnología en general y la aplicada a la información en particular han cobrado un gran peso en múltiples áreas: los negocios, la medicina, la logística, las comunicaciones... y también en el terreno de la educación, donde abre infinitas posibilidades para facilitar y mejorar el aprendizaje de los alumnos así como la capacidad docente de los profesores.

Los sistemas informáticos están cada vez más presentes en los centros docentes (aulas informáticas, aulas de clase con ordenadores, despachos de profesores...) y en los hogares (el número de hogares que tienen al menos un ordenador no deja de crecer). Por ello, desarrollar herramientas software que la comunidad docente pueda emplear tanto en las aulas como fuera de ellas parece un buen complemento a las tradicionales explcaciones del profesorado y un importante apoyo para el estudio personal del alumno.

Son este tipo de cualidades las que pretende aprovechar la línea de investigación y desarrollo abierta en un campo, el de la docencia de algoritmia, en el que aún quedan muchas posibilidades tecnológicas por explorar, como la animación de algoritmos según la técnica de diseño por la que han sido diseñados. La línea de desarrollo abierta con SRec se centra en la visualización de la recursividad.

Hasta ahora, la visualización animada de algoritmos basándose en su técnica de diseño era un campo que, salvo alguna excepción, no estaba apenas explorado y no permitía en el ámbito docente poder acceder de manera sencilla y poco costosa a ejemplos que permitieran entender tales algoritmos.

Por un lado, los profesores necesitaban emplear un amplio número de horas con el fin de generar una animación no configurable ni interactiva a través de herramientas ofimáticas o de diseño gráfico. Esto exigía un algo grado de conocimiento sobre estas herramientas, que en muchas ocasiones requería cursos formativos previos o periodos de aprendizaje independiente que a veces lograban promover el desánimo entre el profesorado y, en consecuencia, la no realización de ejemplos vistosos.

En muchos casos, los profesores no veían otra alternativa que desarrollar ejemplos sencillos, con un diseño limitado, a menudo estáticos, sin posibilidad apenas de configuración o adaptación a los requerimientos de la clase, y que no expresaban

toda la información que se puede transmitir a través de una animación interactiva y totalmente configurable.

Por otro lado, los alumnos se encontraban hasta ahora con representaciones que mostraban ejecuciones completas, gráficos que solían contener indicaciones (flechas, textos, etiquetas...) que intentaban salvar las deficiencias de una representación estática indicando secuencias, significados, relaciones... pero que terminaban enredando la representación, convirtiéndola en una visualización menos clara y más árida para el estudio.

Además, no disponer de alguna herramienta como SRec les obligaba a limitarse al estudio de los ejemplos expuestos en clase o en libros, no teniendo la posibilidad de generar nuevas visualizaciones animadas para otros ejemplos no entendidos, ejercicios propios o prácticas propuestas. Esta situación mermaba el proceso educativo, dificultándolo enormemente para ambas partes, profesorado y alumnado, impidiendo así comprobar y mejorar la eficacia de las visualizaciones como elemento docente.

El funcionamiento y manejo de SRec es muy intuitivo. El resultado de la ejecución del algoritmo se representa mediante un árbol de activación (vista principal de las cuatro que ofrece) que va mostrando, a medida que se avanza en la ejecución, las diferentes llamadas recursivas que se han ido generando, así como los valores de retorno de las mismas. Según avanza la animación de la visualización, la pila de control (segunda vista) va mostrando las llamadas que aún no han finalizado su ejecución.

SRec ofrece dos vistas adicionales, la traza, que refuerza la información sobre las diferentes llamadas recursivas que se realizan sobre el algoritmo y la vista de código, que expone de manera estática el código del algoritmo para comprender mejor su funcionamiento según lo que se va observando en el resto de vistas del programa.

La utilización de los colores ha sido uno de los elementos clave para permitir a SRec adaptarse a las necesidades de cada momento. Así, los colores pueden ayudar a diferenciar partes del árbol de activación que ya han sido totalmente resueltas y también a adecuarse al entorno en el que se está usando el programa, ya sea una clase con media luz a través de un proyector o un espacio más pequeño haciendo uso de un monitor personal.

De esta manera, las tecnologías de la información muestran mediante animaciones la ejecución de algoritmos, permitiendo ver de una manera más clara y desarrollada que a través de libros y representaciones estáticas la recursividad. Éstas ofrecen a menudo la vista total de la ejecución pero no las transiciones paso a paso, como sí permite ver el programa aquí presentado.

El lenguaje en el que deben introducirse los algoritmos en el programa es Java. Se ha elegido este lenguaje por ser ampliamente utilizado tanto en entornos académicos como empresariales y resultar fácil de aprender una vez se tiene una mínima experiencia con otros lenguajes.

En definitiva, SRec pretende sacar partido a las tecnologías más extendidas, aportando una funcionalidad de gran utilidad para profesores y alumnos en el ámbito docente informático.

2. Objetivos

SRec tiene como principal objetivo minimizar el esfuerzo que supone para los profesores el desarrollo de visualizaciones recursivas empleando otras herramientas complejas de carácter ofimático o de diseño gráfico, que no aportan facilidad específica alguna para la creación de visualizaciones ni mucho menos para la generación de animaciones configurables *a posteriori*.

Si bien es cierto que existen otros sistemas de animación de la recursividad, en muchos de ellos suele resultar muy costosa la generación de nuevas visualizaciones. La diferencia que instaura SRec es el alto grado de automatización y la mejora de la interacción con el usuario, que facilita la generación de las visualizaciones reduciendo todo el proceso a un pequeño número de clics de ratón.

De esta manera, se potencia la creación de animaciones para su uso durante la impartición de las clases magistrales, enriqueciendo la exposición del profesor y ayudando al alumno a comprender más fácilmente el funcionamiento del algoritmo según la técnica de diseño por la cual fue escrito.

Permitir que el alumnado pueda generar de una forma sencilla nuevas animaciones es otro de los principales objetivos que tiene el programa presentado, permitiendo así dar respuesta a la demanda del alumnado de poder visualizar cualquier ejercicio o práctica en cualquier momento (clases prácticas en laboratorio, realización de ejercicios en casa, repaso de los ejemplos presentados en clase...).

Se logra contar así con una aplicación orientada a la visualización de la recursividad a través de la generación automatizada sin esfuerzo de visualizaciones. Esto permite alcanzar el objetivo propuesto de facilitar el estudio de los algoritmos basándose en la recursividad empleando para ello una herramienta software sencilla, adaptable y de carácter genérico.

2.1. Trabajos previos

Antes de comenzar a desarrollar el programa, se vio la necesidad de llevar a cabo dos búsquedas de información de origen y tipo completamente diferente.

Por un lado, partiendo de la hipótesis de que las visualizaciones más útiles basadas en técnicas de diseño de algoritmos son algunas de las más empleadas por los autores, se realizó una búsqueda por parte de D. Luis Fernández Muñoz, Profesor de la Escuela de Informática de la Universidad Politécnica de Madrid, de visualizaciones en un amplio catálogo bibliográfico, analizando el uso que realizaban los autores de estas visualizaciones para ilustrar los algoritmos que aparecían explicados en la obras [3].

En la mayoría de casos se observó que el empleo de visualizaciones dependientes del dominio (como la que aparece en la ilustración nº 1) predominaba frente al uso de visualizaciones genéricas, menos expresivas pero que pueden dar mucha más información sobre la técnica de diseño empleada.

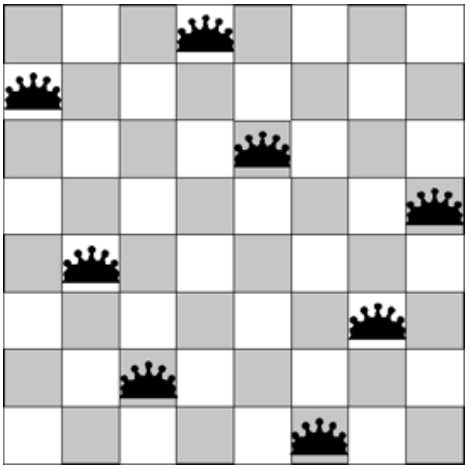

Ilustración 1 Problema de N-reinas

Por otra parte, se realizó en el curso 2004/2005 una encuesta a los alumnos de la asignatura “Diseño y análisis de algoritmos” de la carrera Ingeniería de Informática para conocer el grado de aceptación de las visualizaciones propuestas (árbol de activación y grafo de dependencia para la eliminación de la recursividad; árbol de activación y definición inductiva para la técnica de divide y vencerás; árbol de búsqueda para *backtracking*; grafo de dependencia, tabla de valores y tabla de decisiones para programación dinámica).

El resultado se caracterizó, como era de esperar en parte, por la buena acogida en general que tuvieron las visualizaciones entre el alumnado, que reclamó un uso más generalizado de las mismas en ejercicios y ejemplos, no sólo durante las exposiciones teóricas del profesor.

Partiendo de estas premisas, parece claro que la solución al problema originalmente planteado pasa por la creación de un sistema software de visualización de algoritmos, basado en las técnicas de diseño, con un carácter genérico, y que proporcione amplias opciones de configuración y adecuación a las necesidades que tengan en todo momento profesores y alumnos.

Así, se creó, en colaboración con Don Luis Fernández Muñoz, un primer prototipo que implementara las opciones básicas de navegación para poder tener una primera idea del interés real que tiene la línea de investigación abierta.

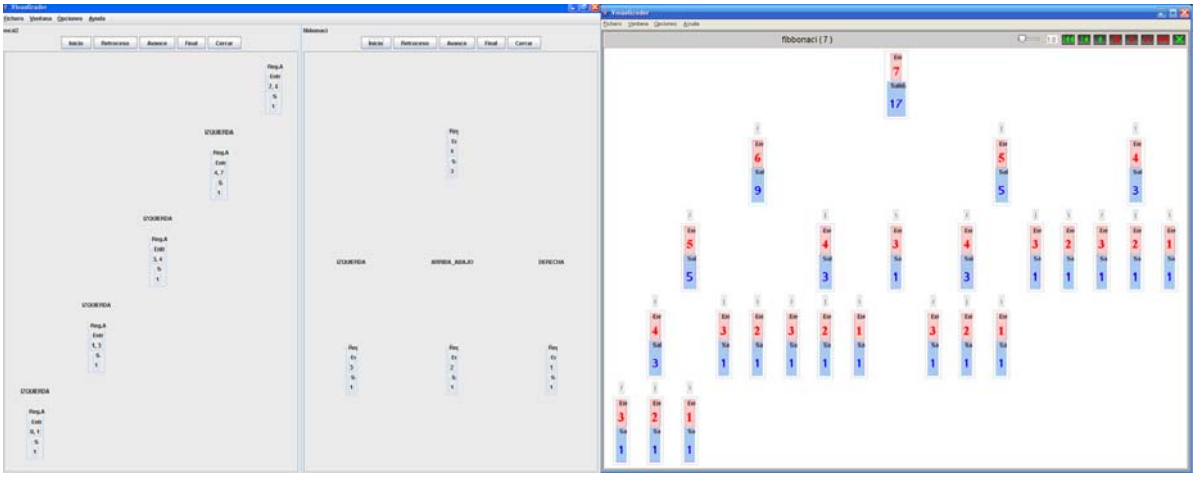

Ilustración 2 Prototipos intermedios de SRec

En este punto se comenzó a trabajar desarrollando de una manera iterativa e incremental sucesivos prototipos, como los que aparecen en la ilustración nº 2, que iban acumulando mayores opciones de configuración y procesado de algoritmos hasta llegar al programa que se presenta, tras año y medio de trabajo.

2.2. Objetivos detallados

Como ya se ha comentado, SRec tiene como objetivo fundamental la reducción de esfuerzo en las tareas necesarias para la generación de animaciones sobre la recursividad. Esto favorece la tarea del personal docente, facilitando el enriquecimiento de las clases magistrales, y del alumnado, que podrá contar con una herramienta que le facilite la comprensión de los ejemplos expuestos en clase, la depuración de prácticas, el estudio de ejercicios, etc.

- El programa debe generar con un esfuerzo mínimo cada una de las visualizaciones que el usuario estime oportuno crear y utilizar en cada momento.
- Las tareas de mantenimiento (almacenamiento y recuperación de visualizaciones guardadas previamente) deben ser igualmente sencillas e intuitivas.
- El programa debe ofrecer varias vistas complementarias de la recursividad, lo que redundará en una mayor expresividad de la información mostrada potenciando además la interacción del usuario.
- El programa deberá ser flexible en cuanto a su configuración, permitiendo la adaptación del formato gráfico de las vistas (colores, formas, etc.), el tamaño relativo de las mismas respecto al tamaño de la ventana (mediante tiradores), la cantidad de información que se muestra (parámetros, valor de retorno, todos, tratamiento de información histórica, nodos según el método al que pertenecen...), etc.
- El programa también debe aportar una solución a los procesos recursivos de gran tamaño, que suelen resultar muy costosos, o directamente inviables, de construir en otros sistemas, pero que en SRec apenas toman unos pocos segundos. El programa ofrecerá facilidades, como un visor, para la navegación por árboles de gran tamaño.
- SRec también tendrá como misión ser capaz de exportar a formatos estándar las visualizaciones que genera, por un lado en formatos gráficos (PNG, GIF, GIF animado, JPG) y por otro en lenguajes de marcado (XML).
- SRec podrá exportar a formato HTML el contenido de la vista de traza.
- SRec ofrecerá información detallada sobre las animaciones (número de nodos y estadísticas diversas) y sobre nodos concretos (número de hijos y subhijos, estado...).
- Para conseguir cumplir los objetivos anteriores, será necesario que la interfaz del programa sea fácilmente usable, tenga una apariencia intuitiva y un comportamiento coherente.
- Tras la implementación del programa, un importante objetivo pasará por la difusión del programa con el fin de conseguir su aceptación entre la comunidad educativa, fomentando su uso y conocimiento. En ello se han realizado esfuerzos desarrollando una página web propia para SRec y dotando a éste de un sistema multilingüe que permite ser utilizado en español e inglés.

2.3. Tecnología de desarrollo

Son varias las tecnologías empleadas a lo largo del proceso de implementación de SRec. Éstas se comentan a continuación.

2.3.1. Java

SRec ha sido programado empleando el lenguaje Java diseñado por Sun Microsystems por varios motivos:

- facilidad para desarrollar interfaces gráficas
- portabilidad entre plataformas
- facilidad para el encuentro de librerías externas que realicen tareas puntuales
- proximidad para el manejo de XML y transformaciones entre Java y dicho lenguaje de marcado

La programación de interfaces gráficas suele ser uno de los puntos que más tiempo de desarrollo consumen a lo largo del proceso de creación de software.

Ello está causado por varios factores, como la necesaria complejidad de la interfaz, que puede no limitarse a mostrar datos por pantalla ya que además suele ser común que ésta recoja la interacción del usuario, dándole la posibilidad de introducir datos, seleccionar opciones, cancelar operaciones, etc. Esto exige un nivel de coordinación alto entre la parte de interfaz y la de la lógica de control que maneja la aplicación para la captura de eventos y la posterior acción en consecuencia.

La creación de ventanas, cuadros de diálogo, cuadros de información, áreas de visualización, etc. resultan en Java más fáciles de programar que en otros lenguajes, debido en gran parte a los paquetes AWT y Swing que incorporan las distribuciones del lenguaje publicadas por Sun Microsystems.

Además, al ser la interfaz la parte de la aplicación con la que interacciona el usuario final, suele ser recomendable que éste acepte y asimile totalmente la interfaz ofrecida por la aplicación, por lo que el desarrollo de la misma, en general, puede hacerse demasiado largo en el tiempo y, consecuentemente, demasiado caro.

En SRec, la interfaz ha sido una de las partes más cuidadas y a la que más atención se ha ofrecido, ya que esta aplicación está orientada a la visualización de información, siendo la interfaz su vía de difusión.

Por todo ello, Java representa una de las mejores opciones en este proceso de creación de software, donde en todo momento se ha ido generando de manera incremental una interfaz.

Dentro de Java se ha realizado un amplio uso del paquete Swing por ofrecer una amplia gama de componentes que permiten implementar interfaces gráficas que ofrecen la posibilidad de recoger datos del usuario (necesario para el uso de cuadros de selección, la introducción de valores para los parámetros de las llamadas principales, la selección de valores para las opciones de configuración...).

Otra de las características muy importantes de Java es la portabilidad de las aplicaciones entre distintas plataformas, si bien es una posibilidad que no ha sido explotada hasta el momento. Teniendo en cuenta que algunos de los estudiantes de informática suelen ser usuarios de plataformas como Linux, podría ser interesante contar con una aplicación que permitiese trabajar tanto en Windows, el sistema instalado en los ordenadores de clase y de las distintas aulas de laboratorio, como en Linux, cada vez más popular entre la comunidad estudiantil.

Desde un primer momento se tuvo clara la idea de que sería necesario contar con una librería avanzada externa que permitiera dibujar árboles con un diseño agradable, estilizado y atractivo. Para los intereses del programa, las librerías estándar de Java se quedan cortas, por lo que se inició una búsqueda de librerías gráficas que ofrecieran ciertas características.

Tras dicha búsqueda, se vio que la mayoría de las librerías estaba implementada en el lenguaje de Sun Microsystems, por lo que la elección de Java como lenguaje de implementación estaba ya plenamente justificada.

Además, la implementación de las interfaces de DOM y SAX, necesarias para el manejo de documentos XML, terminó de favorecer a Java como lenguaje para el proceso de implementación de SRec.

DOM (Modelo de Objetos del Documento) aporta un conjunto de interfaces que permiten manejar los documentos XML de una manera genérica a la vez que intuitiva.

De esta manera, resulta muy sencillo navegar por los documentos XML, así como extraer información de los mismos, modificarla, eliminarla, ampliarla, etc. SAX (Simple API por XML) también presenta una interfaz de programación pero no ha sido utilizada directamente en SRec, aunque sí a través de DOM.

Pese a que hay otros lenguajes que implementan los *parsers* definidos por DOM y SAX, como C, finalmente se escogió Java por todos los motivos expuestos anteriormente.

2.3.2. Swing

Swing es la librería integrada de facilidades gráficas que proporciona Java desde su versión 1.2 para la creación de interfaces gráficas sin exigir un alto esfuerzo para el programador. La librería es independiente de la plataforma, por lo que cuenta con una interpretación universal de los *widgets* que proporciona.

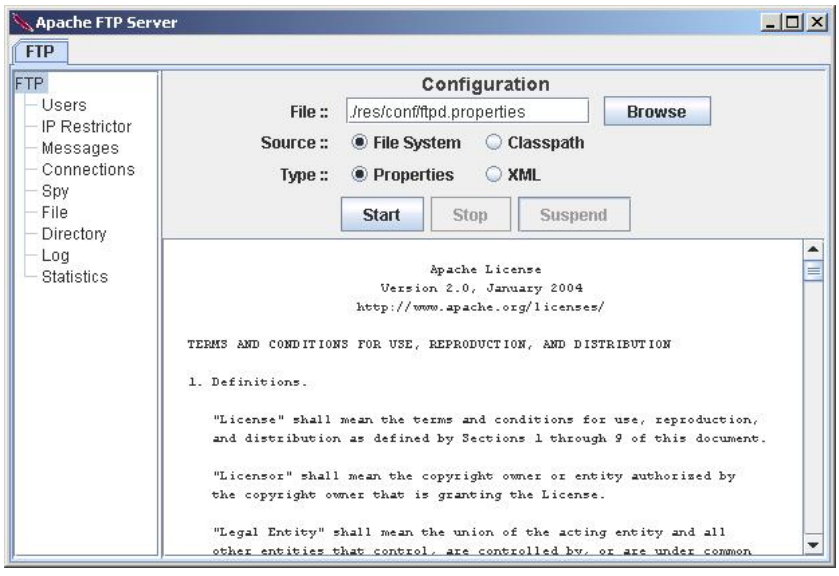

Ilustración 3 Aplicación Apache programada en Swing

Swing ofrece una arquitectura particionada y abierta, que resulta fácilmente extensible por parte de los programadores para matizar y adaptar los elementos gráficos que ofrece a las necesidades propias. Esta característica será fundamental para la integración en SRec de la librería JGraph, como se comentará más adelante.

La librería permite la aplicación de diferentes *look and feel* que ofrecen la posibilidad de que los diferentes elementos gráficos tomen la apariencia estándar del sistema en el que se está ejecutando la aplicación o bien tomen una apariencia

completamente diferenciada, según las necesidades del programador o usuario. En la ilustración 3 aparece una aplicación programada empleando el *look and feel* estándar de Swing.

La principal ventaja de Swing es que su empleo no está restringido por limitaciones de la plataforma al estar programada totalmente en Java, sin hacer uso directo de las herramientas del sistema. Esto además provoca que los componentes ofrezcan funcionalidades más completas.

Sin embargo, el principal inconveniente que tiene que la librería esté programada íntegramente en Java es que su funcionamiento es más lento que el de los elementos nativos. Además, puede no ofrecer el mismo aspecto en todas las plataformas.

2.3.3. JGraph

Es un proyecto de código abierto y permite su utilización gratuita con fines académicos, según informan en su página web http://www.jgraph.com. JGraph proporciona una librería gráfica de carácter avanzado programada en Java y que permite una fácil integración son Swing gracias a la extensión que realiza de su jerarquía de clases. Es compatible con todos los requisitos que se exigen para la ejecución de programas en Java y asume el patrón Modelo Vista Controlador.

JGraph da la posibilidad de crear fácil y cómodamente todo tipo de grafos, permitiendo múltiples opciones de configuración en cuanto a formato referidas al tipo de fuente, a los tamaños de los elementos, a los colores de los mismos, a la modificación de celdas (texto, posición, tamaño, opciones de edición…), etc.

JGraph ofrece algunas características de nivel avanzado tales como manejo de capas, agrupaciones de elementos, manejadores, recuperación de acciones, herramientas Bezier, así como muchas otras que escapan de las necesidades de SRec.

Esta librería es fácilmente manejable gracias a la API que proporciona, que da todo tipo de facilidades fácilmente detectables gracias a la documentación con la que cuenta. Toda la información sobre esta librería se puede encontrar en la dirección web indicada anteriormente y también en el apartado de bibliografía.

2.3.4. Lenguaje XML

XML (eXtensible Markup Language) es el lenguaje de marcado más popular y que más divulgación ha tenido en los últimos años como lenguaje estándar de intercambio de datos de manera estructurada y fácilmente. Está desarrollado por el World Wide Web Consortium desde una simplificación del lenguaje SGML.

Este metalenguaje permite definir mediante una DTD o un *XML-schema* una jerarquía de elementos que pueden representar todo tipo de información estructurada. Esta característica le hace muy útil para adaptarse a todo tipo de programas (bases de datos, editores de texto, hojas de cálculo, diseño asistido…) y de información, convirtiéndolo en una de las herramientas más empleadas en Internet en los últimos años.

Mediante la generación de etiquetas, que albergan cierta semántica dependiente del contexto, se puede extender el lenguaje para representar la información que se quiere transmitir o almacenar. La creación de etiquetas nuevas no implica que se deba implementar un analizador básico nuevo, por lo que se el esfuerzo se puede centrar en el desarrollo de la aplicación que maneja la información contenida en el documento XML y no en el procesamiento del documento.

Por todo ello, XML es ideal para potenciar la compatibilidad entre las aplicaciones, al ser fácil de leer y de comprender.

2.3.5. Formato GIF

El formato GIF (Graphics Interchange Format) fue desarrollado por CompuServe. Define un formato gráfico de archivo que permite contener imágenes de hasta 256 colores. Está ampliamente extendido por la red debido a dos factores:

- para la visualización de gráficos sencillos permite ofrecer archivos de poco tamaño sin pérdida alguna de calidad (siempre que la imagen de origen tenga un número de colores menor o igual a 256).
- permite la inclusión de animaciones en un único archivo (versión GIF89a), visibles en todos los navegadores web sin necesidad de *plugins* o parches añadidos así como en visores de imágenes avanzados.

El formato GIF cuenta con dos versiones: la primera de ellas, GIF87a, data de 1987 y ofrece compresión LZW (sin pérdidas) y soporte para el almacenamiento entrelazado; dos años después culmina su revisión y ve la luz la segunda versión, GIF89a (mucho más extendida), que añade la capacidad de mostrar píxeles transparentes y animaciones. De esta forma, el formato GIF supone una herramienta muy importante para ayudar en la difusión de visualizaciones generadas por SRec a través de Internet, ya que no será necesaria la instalación del programa para poder verlas (si bien es cierto que este formato no permite ningún tipo de configuración ni modificación de la información contenida).

2.4. Metodología de desarrollo

En cuanto a la metodología de desarrollo, SRec ha sido implementado siguiendo un proceso evolutivo e incremental. Así, se recibían periódicamente nuevos requisitos, y se realizaba posteriormente a dicha recepción un proceso de análisis y diseño que terminaba desembocando en una nueva versión del programa que añadía una nueva funcionalidad a las ya existentes. Se aporta una visión esquemática de esta metodología en la siguiente ilustración.

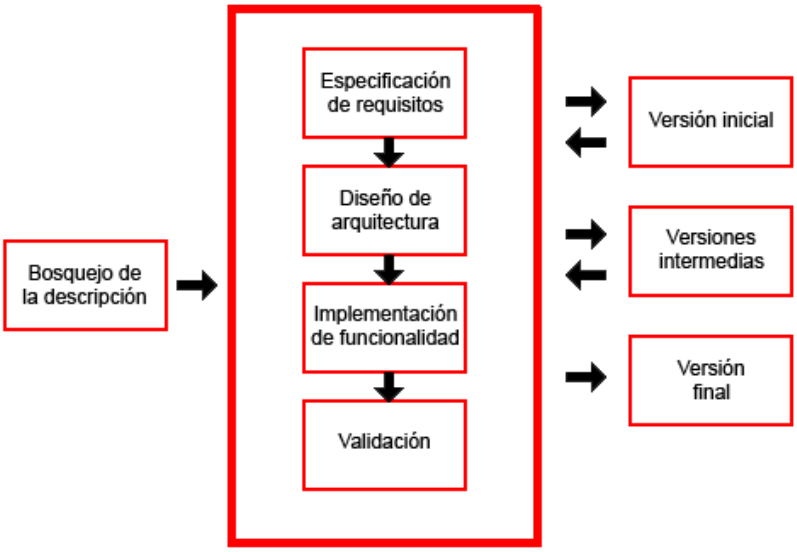

Ilustración 4 Modelo de desarrollo

Durante el desarrollo de algunas funcionalidades se valoraban posibles alternativas, seleccionando siempre la que más se ajustaba a los requisitos, y valorando la capacidad de reutilización que ofrecía (de ahí que algunas herramientas se hayan implementado con un carácter genérico, fomentando así su potencial capacidad de reutilización).

En muchos casos, los nuevos requisitos matizaban los requisitos iniciales, aunque en otras los cambiaban totalmente, orientando la aplicación hacia un camino diferente al establecido de manera inicial. Esto está debido en gran parte a que a medida que la aplicación se iba desarrollando, se realizaban pequeñas pruebas que daban pistas sobre los puntos fuertes y débiles de la aplicación.

Se aprovechaban por tanto dichas pruebas, de carácter interno, para potenciar o matizar ciertos aspectos de la aplicación referidos fundamentalmente a la interfaz y la funcionalidad ofrecida, con el fin de obtener una aplicación lo más adaptada posible a su objetivo propuesto.

El proceso de creación de software ha sido en algunos momentos una tarea complicada, aunque el hecho de mantener de manera diferenciada y separada los datos y la parte de la visualización e interfaz ha facilitado que los cambios sobre la aplicación para poder acoger nuevos paquetes de clases hayan sido mínimos y totalmente viables.

Desde el principio se ha contado con una arquitectura de clases diseñada bajo el patrón de arquitectura software Modelo Vista Controlador. Este patrón diferencia tres partes en la arquitectura del programa. El modelo representa los datos y su lógica de control; la vista se centra en la interfaz del programa y el controlador se encarga de la interacción con el usuario.

El concepto de modularidad siempre ha estado presente durante el proceso de creación del programa, proporcionando módulos de alta cohesión, sobre todo referido a la cohesión funcional (los paquetes diseñados contienen clases cuya funcionalidad está fuertemente ligada o es muy parecida) y de comunicación (las clases de cada paquete pueden acceder a los mismos datos para manipularlos o realizar tareas de procesado, como las que implementa SRec).

El desarrollo de la aplicación ha seguido las directrices que impone el desarrollo orientado a objetos, tratando los datos y sus procesos asociados de manera conjunta y modularizada. No obstante, algunas clases han sido diseñadas como clases de servicios, buscando un ulterior uso cómodo y de carácter genérico y evitando así tener que conocer en algunos casos la identidad de los objetos, necesidad que impone el estilo arquitectónico orientado a objetos.

3. Sesión de usuario

La sesión de usuario presentará una utilización habitual del programa, explicando todas sus opciones y pasos de funcionamiento. La motivación de esta sesión es guiar al lector en el uso del programa, si bien no es una presentación exhaustiva de las funciones del programa, que quedarán detalladas en los capítulos siguientes. Por otro lado, el programa debe ejecutarse desde una unidad donde existan permisos de escritura.

3.1. Generar una animación

Para generar una animación, se debe procesar la clase que contiene el algoritmo que deseamos visualizar. SRec, entre otras cosas, realizará la compilación de la clase con el fin de que posteriormente se pueda ejecutar dicho algoritmo con aquellos parámetros que nos interese.

El primer paso requiere emplear la opción “Cargar y procesar clase”, reconocible por el icono (), para comenzar el procesado de la clase. SRec nos abrirá el cuadro de diálogo que permite seleccionar la clase que interesa procesar. En esta sesión vamos a emplear la clase “Algoritmos” que aparece en el disco electrónico adjunto. Tras seleccionarla, SRec comienza a analizar la clase, manteniéndola cargada y compilada para posteriores invocaciones de sus métodos. Tras ello, el programa muestra el código de la misma en el primero de los paneles de la ventana.

A continuación, con seleccionar la opción “Nueva animación…” () SRec podrá generar tantas visualizaciones como quiera el usuario del algoritmo de “fibonacci”.

Cada vez que se pulse el citado botón, el programa mostrará un cuadro con los métodos que fueron procesados previamente, similar al de la ilustración. En este caso, la aplicación muestra un cuadro de diálogo que contiene un listado con todos los métodos declarados en la clase cargada.

Ilustración 5 Cuadro de selección del método que se desea visualizar

El usuario debe escoger uno de ellos para visualizarlo pinchando sobre el botón de radio correspondiente. Las casillas de verificación permiten que el usuario elija qué métodos deben permanecer visibles en la animación (quedarán seleccionados en el cuadro de diálogo) y cuáles no (permanecerán sin seleccionar). El cuadro también permite seleccionar y deseleccionar todos los métodos mediante la casilla de verificación situada en la parte superior. Una vez que se ha procedido con la selección, se debe pulsar el botón "Parámetros" del método que lanzará la ejecución del algoritmo para poder introducir los parámetros con los que SRec debe ejecutar el método:

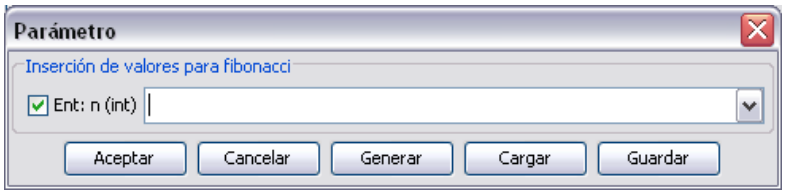

Ilustración 6 Cuadro de inserción de valores para los parámetros

Al introducirlos se tienen varias opciones. La primera de ellas consiste en insertarlos por teclado. Otra opción interesante es generarlos aleatoriamente (botón "Generar"). SRec proporciona un cuadro de diálogo (ilustración 7) que permite escoger los límites inferior y superior así como si los valores, en caso de pertenecer a un array o matriz, deben estar ordenados. Además, se puede dejar al azar el tamaño de los arrays y matrices o bien concretar el tamaño de cada dimensión.

Ilustración 7 Cuadro de generación de valores aleatorios para los parámetros

La tercera de las posibilidades pasa por cargarlos desde un archivo (botón "Cargar"). De esta manera, no es necesario escribirlos ni generarlos cada vez, opción de agradecer en el caso de valores de gran tamaño como por ejemplo matrices. Una opción adicional es emplear el listado desplegable de valores introducidos anteriormente durante la sesión actual, lo que permite cargar los mimos valores una y otra vez sin necesidad de cargarlos desde fichero ni de escribirlos.

Una vez que el valor está insertado y se ha pulsado "Aceptar", el programa ya está en disposición de generar la visualización.

3.2. Manejar una animación

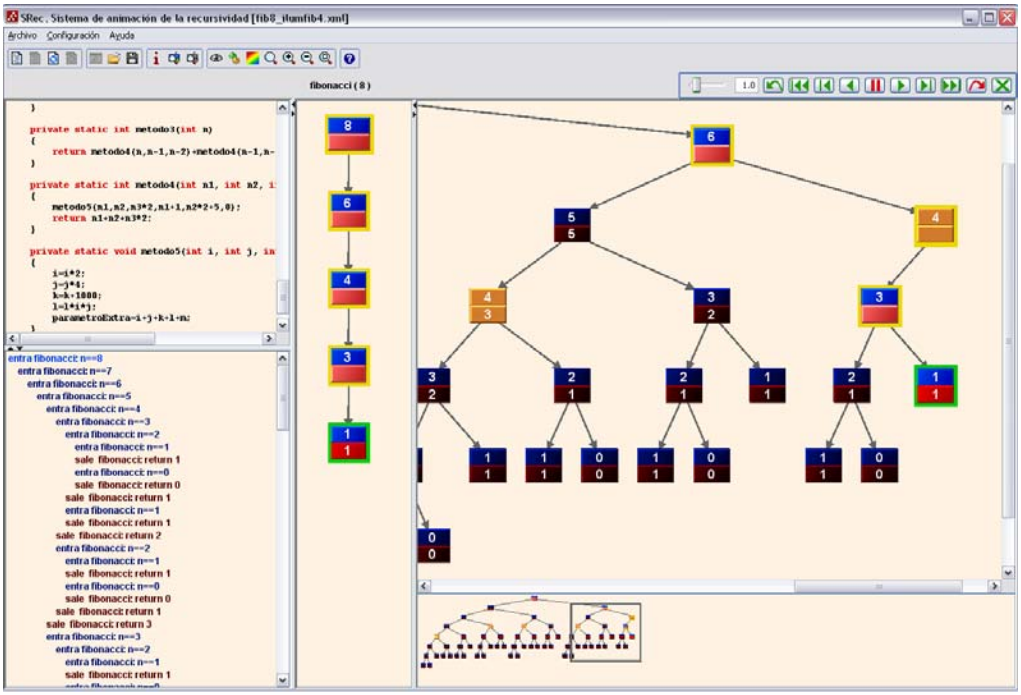

Ilustración 8 Ventana principal de SRec

Hay disponibles cuatro vistas durante el tiempo que una visualización está activa. Tal y como se aprecia en la ilustración 8, de izquierda a derecha aparecen la vista de código (vista estática que contiene el código de la clase procesada), la vista de traza (vista dinámica que mantiene información textual sobre la sucesión de llamadas que se producen y cierran), la vista de la pila de control (situada en el medio de la ventana, ofrece información gráfica sobre las llamadas residentes en la pila) y la vista del árbol de activación (es la más importante y muestra el árbol completo de llamadas). Sobre las vistas aparece en el margen derecho de la ventana una barra de navegación que permite utilizar la animación a conveniencia.

En este punto existe gran libertad de interacción por parte del usuario. Si se desea empezar a emplear las opciones de navegación por la visualización puede activarse la animación pulsando ( ). El botón quedará sombreado en amarillo y la animación se activará: el árbol de llamadas recursivas irá apareciendo en pantalla, la pila de control albergará llamadas nuevas y la traza se completará, todo ello de manera coordinada. La animación se puede detener pulsando ( ).

También se puede navegar manualmente, paso a paso, por la animación mediante los botones ( ) y ( ). Para reiniciar o avanzar totalmente la visualización basta pulsar los botones ( ) y ( ). Además, para realizar saltos sobre determinadas llamadas recursivas se pueden emplear los botones ( ) y ( ).

3.3. Configurar una animación

Por otro lado, se puede configurar la animación a través de las opciones “Formato de árboles…” ( ), “Formato tipográfico….” ( ) y “Configuración de zoom” ( ). El primero de ellos permite elegir ciertas opciones tales como el tipo de información que se ofrece, cómo deben tratarse aquellos nodos que representan información histórica y si deben permanecer en pantalla aquellos subárboles que pertenecen a llamadas sobre las que se ha aplicado la opción de salto.

El segundo de ellos abre un amplio cuadro de diálogo, visible en la ilustración 9, que permite elegir el formato para todos los elementos de la visualización.

Ilustración 9 Cuadro de configuración de formato

Las tres primeras pestañas permiten seleccionar el color de multitud de elementos. La primera de ellas permite configurar el modo 1 de color, que permite distinguir a las celdas que contienen valores de entrada de las celdas que contienen valores de salida. La segunda de ellas contiene las opciones del modo 2 de color, que distingue a los nodos en función del método al que representan. La pestaña 3 da la opción de configurar colores de elementos comunes tales como el fondo de los paneles o el color de las flechas.

Para proceder, se debe seleccionar el elemento sobre el que se desea aplicar el color, después seleccionar el color mediante alguno de los tres métodos que se ofrecen (Muestras, HSB y RGB) y a continuación pulsar sobre “Asignar”. El color quedará asignado automáticamente sobre el elemento y será visible en la ventana con el nuevo color sin necesidad de cerrar el cuadro de diálogo.

La última de las pestañas permite configurar otros aspectos tales como los marcos y bordes de las celdas, la separación a la que se éstas sitúan unas de otras, ciertas características de las flechas, tipos de fuente para las vistas de código y traza… Al seleccionar cada parámetro éste quedará aplicado automáticamente sobre la animación.

3.4. Guardar una animación

Para guardar la animación con el estado y formato actuales, basta seleccionar la opción “Guardar animación…” () y seleccionar la ubicación de destino. A continuación se puede cerrar el programa y volver a iniciarlo, imitando una actuación en el que el usuario desea volver a utilizar SRec en otro momento o lugar. Si el usuario desea volver a trabajar con la visualización generada anteriormente, tan sólo tiene que pulsar sobre el botón () y seleccionar el archivo que generó en la sesión anterior.

Lo que comprobará el usuario (si la vuelve a cargar mediante la opción “Cargar animación…”) es que la visualización queda cargada en el mismo estado, con la misma información y empleando el mismo formato, por lo que estará listo para seguir trabajando en el mismo lugar en el que lo dejó por última vez.

3.5. Editar código

Si el usuario desea depurar el programa o modificar su comportamiento, puede emplear el editor de código de que dispone SRec para ello. Simplemente tiene que seleccionar la opción “Editar clase procesada”, reconocible por el icono ( ), aunque puede hacer uso de la opción “Editar clase…” ( ) si lo que desea es editar cualquier otra clase que deberá seleccionar posteriormente desde el sistema de ficheros.

Una vez que la ventana del editor está abierta, se puede editar libremente el código de la clase. Además, se podrá hacer uso de las funciones básicas de “cortar”, “copiar”, y “pegar”. Lógicamente, el programa brinda la opción de guardar el código modificado así como de abrir una nueva clase para su edición.

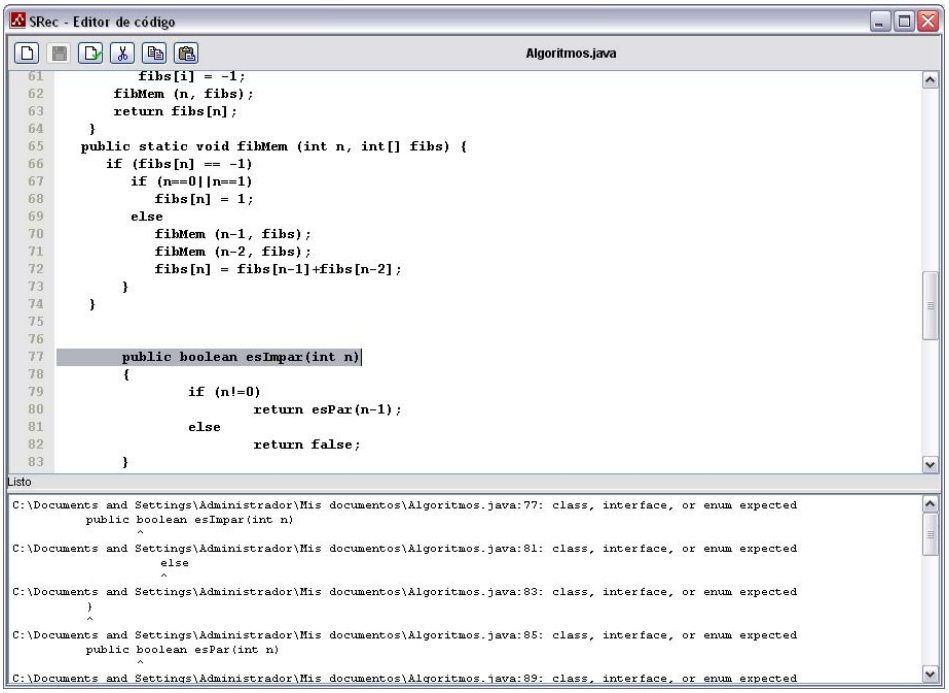

Ilustración 10 Ventana de edición de código

Además, el usuario puede compilar haciendo uso del botón ( ). De esta manera, podrá comprobar fácilmente si los cambios introducidos incluyen algún error que deba ser corregido o no. Si hay algún error, se puede seleccionar la línea donde se notifica para que se señale en el panel superior la ubicación del error. Una vez que se han realizado los cambios, será necesario reprocesar la clase seleccionando en la ventana principal la opción “Reprocesar clase” ( ) para que éstos tengan efecto en las visualizaciones que se generen.

3.6. Información sobre la animación

Mientras permanece abierta la visualización, el programa es capaz de aportar datos variados sobre la misma. Así, si se selecciona la opción "Información de la animación" ( ), aparecerá un cuadro de información que detallará algunos aspectos de la visualización, como el número total de nodos, número de nodos no mostrados aún, número de nodos cuya ejecución ya ha finalizado, etc.

El cuadro, además, ofrece un listado de métodos involucrados en la animación, permitiendo conocer de un vistazo si existe recursividad múltiple en la visualización.

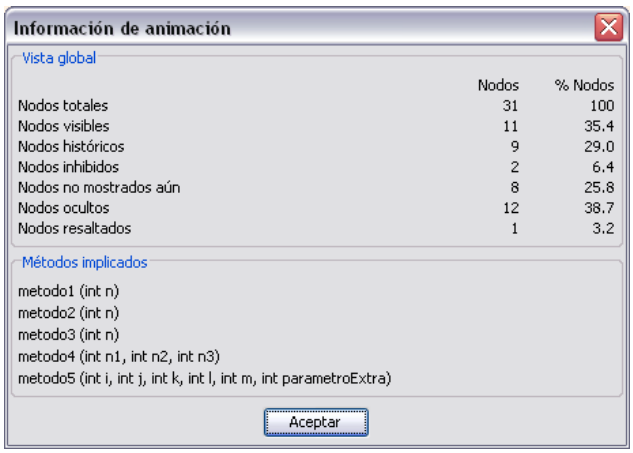

Ilustración 11 Cuadro de información de una animación

3.7. Manual de SRec

Se puede conseguir un manual de SRec en el CD adjunto a este Proyecto Fin de Máster o en la web del Departamento de Lenguajes y Sistemas Informáticos I de la Universidad Rey Juan Carlos de Madrid.

4. Especificación

Dada la metodología de desarrollo empleada, la especificación y el diseño han sido tareas íntimamente ligadas, de ahí que algunas de las cuestiones expuestas a continuación puedan ser consideradas también asuntos propios de la fase de diseño.

La especificación del programa se puede separar en dos grandes áreas: las vistas de la visualización junto con sus acciones de animación y las funcionalidades que aporta el programa para mejorar la experiencia del usuario.

4.1. Formato gráfico de las vistas

SRec proporciona cuatro vistas que ofrecen información complementaria sobre el algoritmo recursivo que se está visualizando. Tres de ellas son de carácter dinámico, la información que muestran va variando según avanza la animación, mientras que la cuarta, que muestra el código que fue ejecutado por SRec previamente, se mantiene invariable a lo largo de todo el proceso.

Con el fin de aumentar el carácter intuitivo de la visualización, las diferentes vistas tienen ligados diversos aspectos de configuración. Así, por ejemplo, se asignan los mismos colores a los mismos datos en las diferentes vistas para lograr una fácil identificación de los mismos.

Se explican a continuación con gran detalle cada una de las vistas que proporciona SRec para cada visualización.

4.1.1. Vista de árbol de activación

Ésta es la vista principal, donde el usuario puede ver de manera gráfica el desarrollo de la ejecución del algoritmo en forma de árbol de activación. Cada nodo representa una llamada recursiva del algoritmo, representando de manera diferenciada los valores de los parámetros de entrada por un lado y el valor de retorno por otro.

Pulsando sobre el nodo, aparece una etiqueta flotante que informa sobre los valores así como sobre el algoritmo al que pertenece esa llamada. Pulsar sobre el nodo con el botón derecho del ratón abre un menú contextual que ofrece cuatro opciones. La primera activa una etiqueta flotante muy similar a la anteriormente citada pero que aporta información más detallada sobre el nodo (valores de parámetros no visibles y nombres de parámetros).

La segunda opción del menú hace que el nodo pase a ser el nodo activo; esto es, el nodo en el cual queda detenida la animación, por lo que la ejecución del citado nodo queda en el estado inicial. La tercera opción muestra un cuadro de diálogo que aporta algunos datos sobre el nodo en cuestión (estado, número de hijos, número de subnodos, método al que pertenece...). La cuarta opción permite resaltar la llamada junto al resto de llamadas que existen en la animación que pertenecen al mismo método y contienen los mismos parámetros de entrada.

Algunos nodos aparecen remarcados con un marco de color. Esto quiere decir que se encuentran almacenados en la pila de control (son llamadas cuya ejecución no ha finalizado). Habrá un nodo que tendrá un color de marco distinto al resto. Será así como quede señalado el nodo activo, el nodo que está siendo tratado en el estado en que se encuentra la visualización en ese momento.

Esta vista, resaltada en la ilustración 12, contiene un visor de posición que permite situarse fácilmente sobre árboles extensos. Con un solo clic, la parte superior se situará allá donde pinche el usuario dentro del área del visor. Lo mismo sucederá si el usuario decide arrastrar el marco del visor: la parte superior quedará ubicada justo donde dicte el marco. Para mantener la coherencia, si se navega con las barras de *scroll* de la parte superior, el marco del visor se desplazará convenientemente, reflejando así por qué zona del árbol está navegando el usuario.

Este visor mantiene la coherencia con la vista superior en todo momento, también incluso si se modifican los valores de zoom o el tamaño de la ventana.

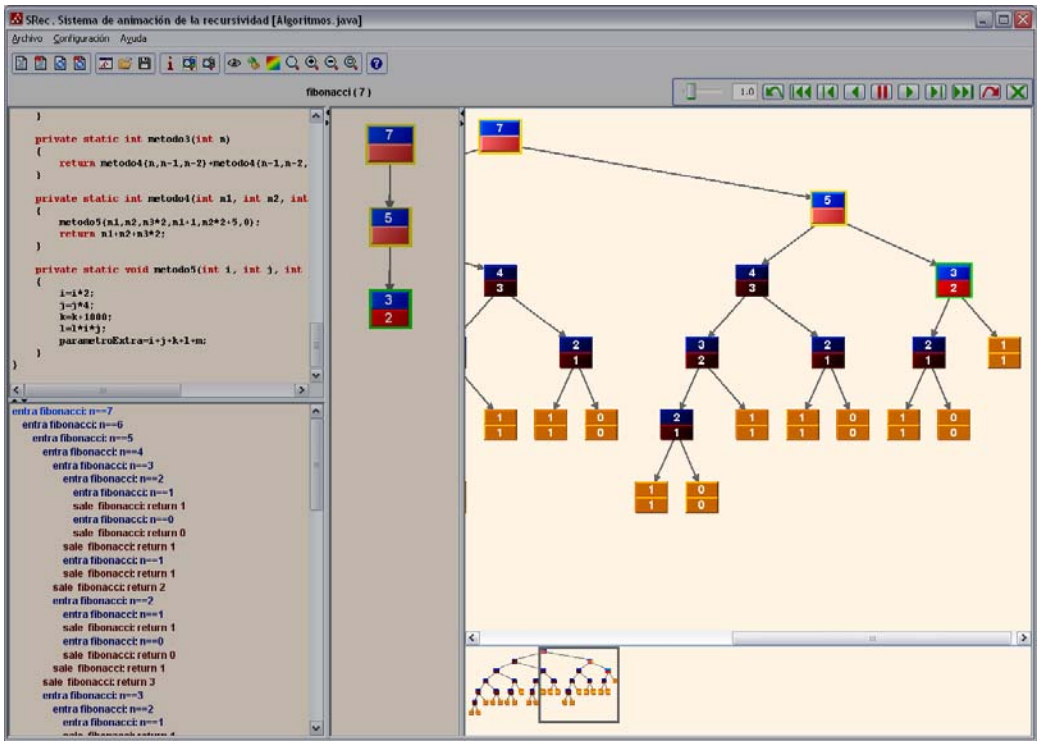

Ilustración 12 Vista de árbol de activación

Los árboles que muestran llamadas a varios métodos diferentes ofrecen un acrónimo precediendo a los valores de los parámetros de entrada con el fin de distinguir las llamadas que se producen sobre uno y otro método. Pulsando sobre esos nodos se podrá ver en la etiqueta informativa el nombre completo del método así como los valores de los parámetros de entrada y de salida.

Según la configuración escogida, los árboles pueden aparecer enteros, con una parte atenuada (nodos históricos) o sólo con la rama actual visible.

Esta vista, que es capaz de ofrecer el árbol de activación al completo (siempre y cuando la configuración elegida por el usuario así lo dicte), ayuda a valorar la eficiencia en tiempo del algoritmo, ya que se pueden realizar estimaciones por cada una de las llamadas que aparecen en esta vista.

4.1.2. Vista de pila de control

Esta vista ofrece una visión independiente de la pila de control, que contiene todas las llamadas que se encuentran insertadas en ella pendientes en cada momento. Estos nodos coinciden con los que aparecen remarcados en la vista del árbol de activación.

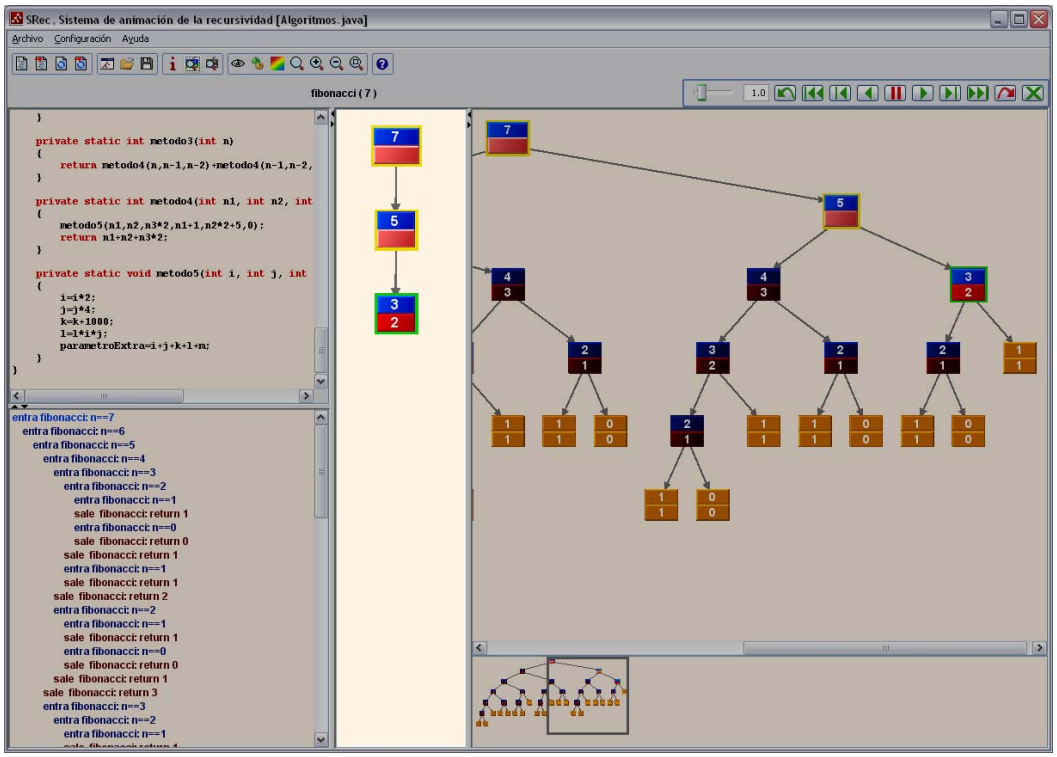

Ilustración 13 Vista de pila

La vista de la pila, resaltada en la ilustración 13, mantiene en todo momento el mismo formato (colores, bordes estilizados de nodos...) que la vista del árbol de activación. Esto favorece la identificación y la comprensión de la información que se ofrece, en complemento a la vista del árbol.

Esta vista facilita el análisis de la eficiencia en espacio, ya que ofrece en todo momento el número de llamadas que se mantienen residentes en memoria mientras no finalicen su ejecución.

4.1.3. Vista de traza

Gracias a esta vista, el usuario dispone en formato textual de un historial de las llamadas que se han realizado hasta el momento actual en que se encuentra la visualización. Esta información se va actualizando de manera sincronizada con el progreso de la animación.

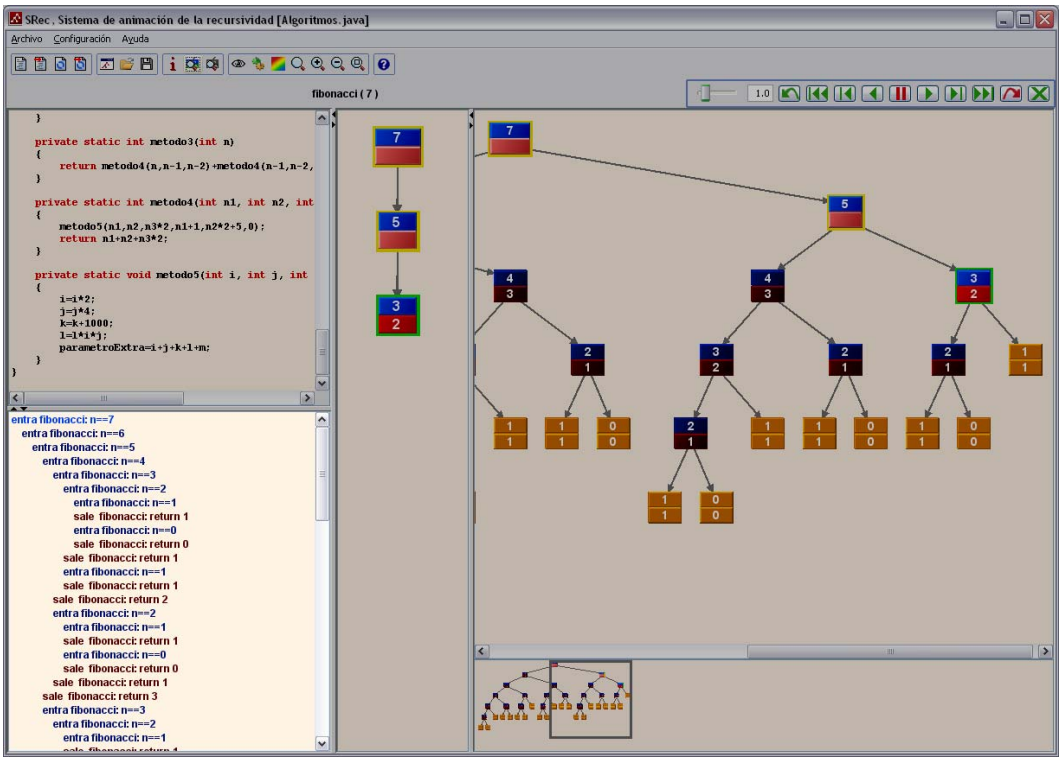

Ilustración 14 Vista de traza

Imita el comportamiento de la vista principal, el árbol de activación, omitiendo llamadas finalizadas si se ha configurado SRec para que la información histórica no aparezca en la visualización o mostrándola atenuada si es así como se muestra en la vista del árbol.

En esta vista, que aparece resaltada en la ilustración 14, aparecen los nombres de los parámetros en las líneas que representan las llamadas recursivas, para facilitar la comprensión de la ejecución del algoritmo recursivo.

4.1.4. Vista de código

La vista de código, la única vista estática de las cuatro que ofrece el programa, permite ver el código de la clase que contiene el algoritmo que se está visualizando. Aparece resaltada en la ilustración 15.

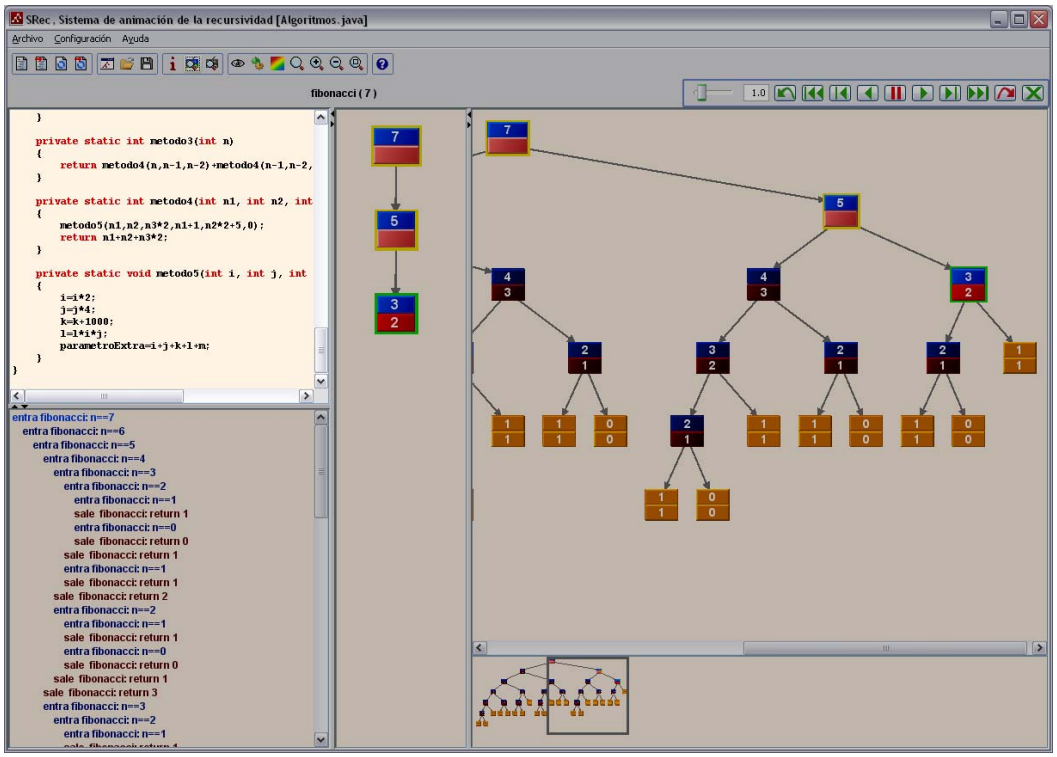

Ilustración 15 Vista de código

Se presenta escrito con múltiples colores, para diferenciar las palabras reservadas del lenguaje, los comentarios y el nombre del algoritmo (logrando así una fácil identificación del método en concreto y de las llamadas recursivas).

Este panel se puede configurar por parte del usuario en cuanto a colores, tamaño y tipo de fuente que se emplea.

4.2. Funciones de la aplicación

Más allá de las vistas que proporciona el programa sobre la visualización de la recursividad, SRec contiene una gran multitud de funciones que facilitan su manejo, permiten gestionar su configuración y en definitiva, adecuar el programa a las necesidades del usuario.

A continuación se lista una serie de funcionalidades, algunas de ellas muy importantes para el funcionamiento ordinario del programa.

4.2.1. Control de una animación

La navegación por la visualización presenta múltiples opciones, disponibles a través de la barra de navegación:

Ilustración 16 Barra de animación

- Pausa (quinto botón): en este estado la visualización permanece estática mostrando un estado concreto de la animación. Es el estado en el que se inician las visualizaciones.
- Paso hacia delante (séptimo botón): Avanza el estado de la animación un paso, abriendo una nueva llamada o cerrando una llamada que acaba de finalizar.
- Paso hacia detrás (tercer botón): Retrocede el estado de la animación un paso, dejando inactiva la última llamada pendiente o dejando abierta una llamada que ya estaba finalizada.
- Ir al principio (segundo botón): Accede de manera directa al estado inicial de la visualización, con la primera llamada al método recursivo activa.
- Ir al final (octavo botón): Accede de manera directa al estado final de la visualización, donde todas las llamadas están finalizadas y se conoce el valor de retorno final de la llamada recursiva principal.
- Salto hacia delante (noveno botón): Sólo se puede realizar cuando la llamada activa no está finalizada. En ese caso, se realizará un salto hasta conseguir obtener su valor de retorno. Todas las llamadas dependientes, ubicadas en un subárbol, quedarán ya resueltas igualmente. Según la configuración determinada por el usuario, el árbol de subllamadas podrá mostrarse o no en la vista.
- Salto hacia atrás (primer botón): Sólo se puede realizar cuando la llamada activa está finalizada. En ese caso, se realizará un salto hasta situar la animación en el estado inicial de esa llamada. Ninguna de las llamadas dependientes, ubicadas en un subárbol, habrá sido llamada aún.
- Animación hacia delante (sexto botón): Activa el estado de animación hacia delante, por lo que, de manera periódica, SRec irá avanzando los estados de la animación hasta alcanzar el final. El periodo es regulable.
- Animación hacia detrás (cuarto botón): Activa el estado de animación hacia detrás, por lo que, de manera periódica, SRec irá retrocediendo los estados de la animación hasta alcanzar el estado inicial de la misma. El periodo es regulable.

- Cierre (décimo botón): El programa permite cerrar la visualización en cualquier momento. Cuando el usuario, sin haber cerrado una visualización, quiera iniciar otra, SRec advierte de que abrir una nueva visualización cerrará obligatoriamente la actual, dando la opción de cancelar el proceso.

4.2.2. Almacenamiento de una visualización

SRec da la posibilidad de guardarlas en formato XML con el fin de poder recuperarlas posteriormente para seguir trabajando con ellas en otro momento o lugar. La información que se almacena se divide en tres partes:

- Traza de ejecución, que permite recuperar todo el árbol de llamadas completo
- Estado de la visualización, que brinda la oportunidad de cargar la visualización en el mismo estado de desarrollo en el que fue guardada
- Opciones de la visualización, que ofrece poder mostrar la visualización con el mismo formato y cantidad de información que había cuando fue guardada.

De esta forma, se puede usar una misma visualización tantas veces como se desee sin necesidad de regenerarla cada vez.

4.2.3. Configuración de formato

SRec da la posibilidad de elegir una amplia gama de colores y opciones de formato para los distintos elementos que aparecen en las cuatro vistas de la ventana principal. Todas estas posibilidades son accesibles a través de un cuadro de diálogo dividido en cuatro pestañas que se comentan a continuación.

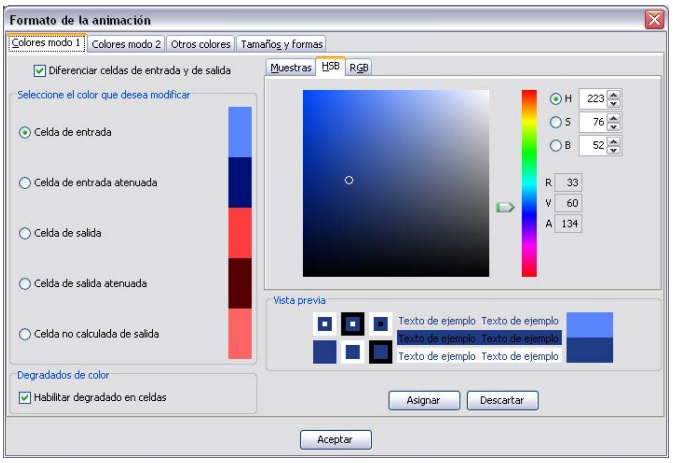

Ilustración 17 Cuadro de configuración de formato (primera pestaña)

La primera pestaña, cuyo contenido se puede ver en la ilustración 17, permite configurar los nodos de las vistas del árbol y de la pila en el modo 1 de coloreado, que distingue con colores diferentes las celdas que muestran valores de parámetros de entrada de las que muestran valores de retorno (o valores de parámetros de entrada tras la finalización de la ejecución correspondiente). La segunda pestaña, mostrada a continuación en la ilustración 18, permite configurar el modo 2 de coloreado, que distingue los nodos del árbol de acuerdo al método al que representan (los nodos de un método quedan coloreados de un color común). Ambas pestañas permiten aplicar efectos de degradados a las celdas de los nodos, así como seleccionar cuál de los dos modos de coloreado se desea emplear en cada momento.

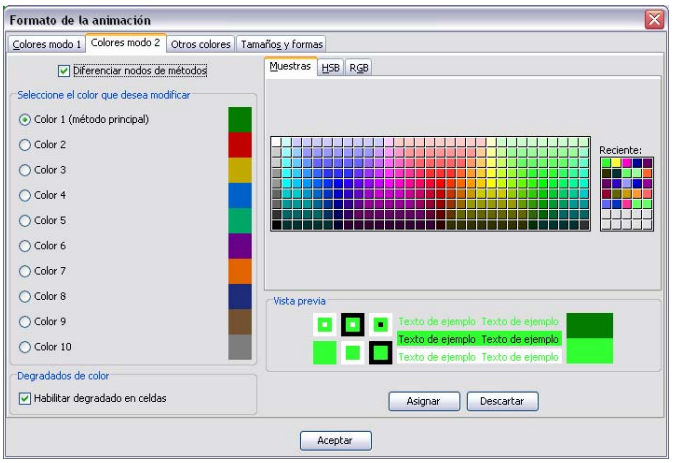

Ilustración 18 Cuadro de configuración de formato (segunda pestaña)

La tercera pestaña permite configurar colores de elementos que son comunes a todas las vistas y que no quedan recogidos en ninguno de los dos modos de coloreado, como el fondo del panel, las flechas, los textos, etc.

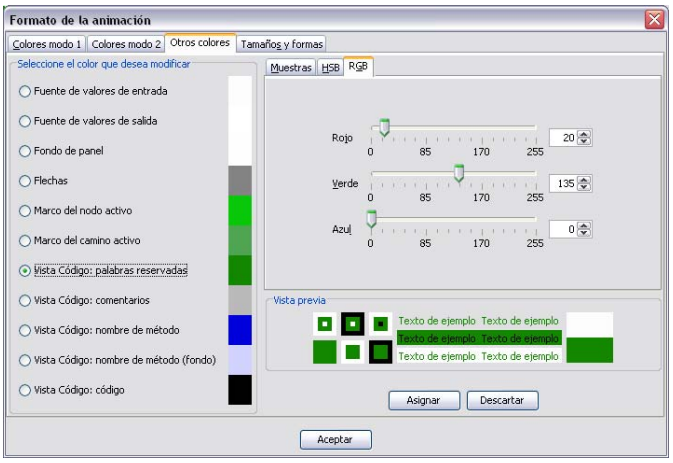

Ilustración 19 Cuadro de configuración de formato (tercera pestaña)

La cuarta pestaña, visible en la ilustración 20, permite configurar otros factores distintos a los colores de los elementos de todas las vistas. Por ejemplo, alberga las opciones de configuración sobre la distancia habrá entre las celdas, qué tipo de borde estilizado utilizarán, qué clase de flechas aparecerán, qué grosor tendrán éstas, qué tipo de letra se empleará en las vistas de código y traza, etc.

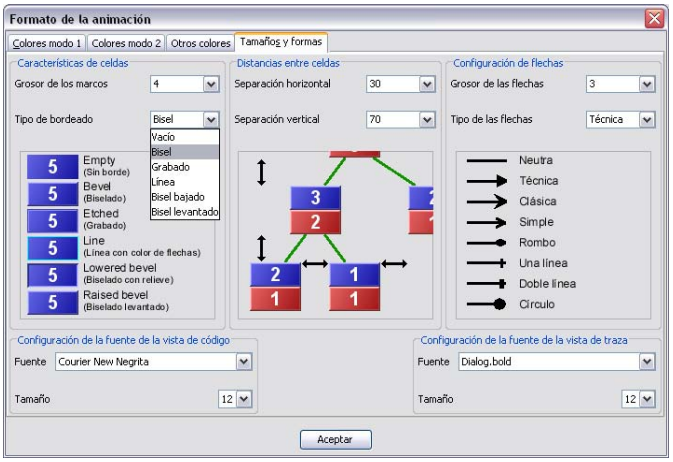

Ilustración 20 Cuadro de configuración de formato (tercera pestaña)

4.2.4. Gestión de visualización

Más allá del formato, las visualizaciones que genera SRec ofrecen diversas opciones de configuración. Por un lado, dan la posibilidad de mostrar todos los datos disponibles en el árbol o bien limitarse a ofrecer en pantalla únicamente los valores de los parámetros de entrada o el valor de retorno de las subllamadas. Esto permite centrarse en una parte de los datos sin necesidad de visualizarlos todos al mismo tiempo, ya que en

cualquier momento se puede cambiar la elección sobre qué información se desea mostrar.

Por otra parte, la visualización puede quedar configurada para que aquellas llamadas recursivas que ya finalizaron y devolvieron su valor (es decir, las llamadas denominadas históricas) se mantengan en la visualización o bien sean eliminadas de la misma. SRec da una tercera opción que consiste en mostrarlas con los mismos colores que las llamadas actualmente activas pero atenuándolas. Así resulta muy fácil distinguir unos nodos de otros.

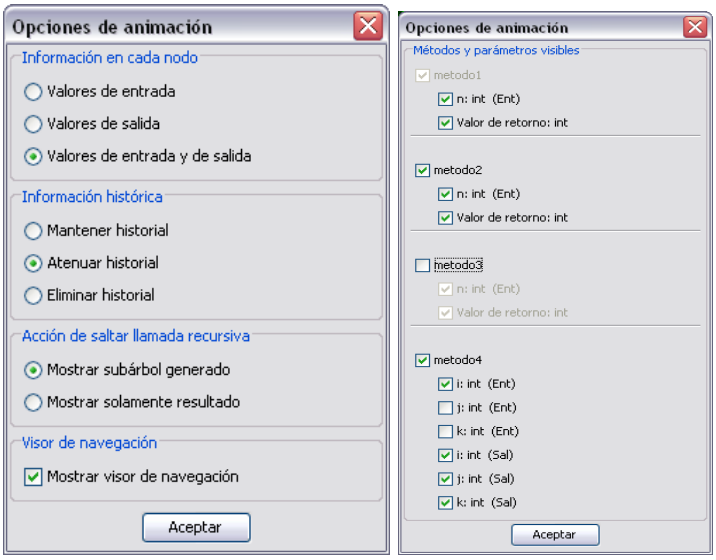

Ilustración 21 Cuadro de opciones y cuadro de visibilidad

Otra opción que incluye SRec es la de mantener en la visualización aquellos subárboles de nodos que no son recorridos sino saltados. La alternativa pasa por suprimirlos de la visualización.

También se dan opciones de configuración para la vista del árbol de activación, ya que es posible hacer desaparecer el visor si éste no aporta facilidad alguna al estar trabajando con árboles pequeños. Todas estas opciones son configuradas mediante el cuadro de diálogo que aparece a la izquierda en la ilustración 21.

Por otro lado, SRec incluye una serie de posibilidades de visibilidad que permiten variar, con la visualización abierta, la cantidad de información que se está ofreciendo. Así, se puede especificar qué nodos de la visualización deben aparecer o desaparecer en función del método al que pertenecen. También se pueden escoger los parámetros que se desean mantener visibles y cuáles hacer invisibles cuando no tienen interés en un momento dado.

En las primeras versiones de SRec estas posibilidades no estaban disponibles, pero se han añadido por la gran flexibilidad que añaden al manejo de la herramienta durante el transcurso de las visualizaciones.

4.2.5. Zoom

Como ya se ha explicado, SRec está destinado a ser utilizado tanto en pantallas de gran tamaño mediante la utilización de proyectores como en monitores individuales, por lo que el tamaño en el que se presentan las visualizaciones tiene que ser fácilmente ajustable a las necesidades de cada momento.

Por ello, SRec permite configurar el efecto de zoom mediante el cuadro de diálogo mostrado en la ilustración 22 con total libertad mediante la introducción de un valor numérico entre -100 y 100. Si el valor es negativo, el árbol se mostrará más pequeño (alejado) que en su estado natural, mientras que si el valor es positivo, el árbol se mostrará a un tamaño proporcionalmente mayor.

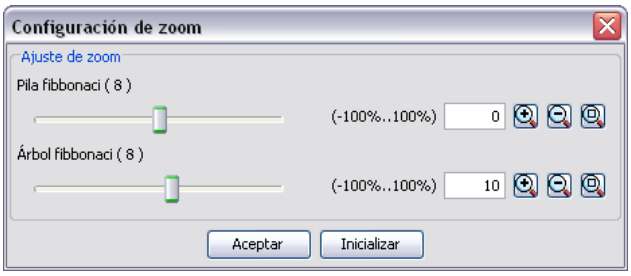

Ilustración 22 Cuadro de manejo de zoom

SRec también da la posibilidad de ajustar el tamaño del árbol al de la ventana mediante la simple pulsación de un botón, lo que ahorra tiempo al intentar ajustar esa característica.

4.2.6. Búsqueda de llamadas

La aplicación permite buscar en la visualización actualmente abierta aquella llamadas que coinciden en método y valores de parámetros con los que nosotros deseamos. Así, mediante el cuadro de diálogo correspondiente, SRec permite escoger entre todos los métodos que están involucrados en la visualización aquel que nos interesa buscar.

En el mismo cuadro de diálogo podemos insertar posteriormente los valores de los parámetros que queremos que coincidan con las llamadas encontradas. Es posible

dejar en blanco el valor de algún parámetro, esto hace que ese parámetro no sea cogido como criterio de búsqueda, por lo que las llamadas sólo deberán respetar el valor del resto de parámetros insertados para ser encontradas. Se podrán seleccionar en los desplegables los valores introducidos con anterioridad durante la actual sesión para buscar nodos o generar animaciones.

Ilustración 23 Cuadro de búsqueda de llamadas

Otra manera de realizar una búsqueda de llamadas es mediante la pulsación del botón derecho sobre un nodo idéntico a los que queremos encontrar. Así, tras seleccionar la opción “Buscar llamadas iguales”, aparecerán resaltadas todas las llamadas que sean iguales en método y parámetros a la llamada seleccionada. Tras realizar la búsqueda, los nodos quedarán resaltados en un color escogido por el usuario a través del cuadro de configuración de formato presentado en el apartado 4.2.3.

4.2.7. Cuadros de información

SRec proporciona cuadros de información que ofrecen información adicional sobre la visualización actualmente abierta y sobre nodos concretos de la misma.

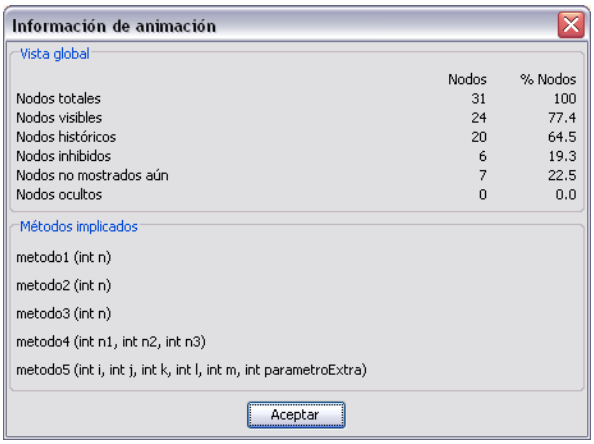

Ilustración 24 Cuadro de información de la visualización

El cuadro de información sobre la visualización se activa desde menú o la barra de herramientas. Éste ofrece datos numéricos sobre la visualización, referente al árbol de activación. Así, se puede saber el número de nodos totales que forman el árbol de activación, el número de nodos que representan llamadas que ya están finalizadas, el número de nodos que permanecen ocultos, el número de nodos que no se han llegado a mostrar aún, el número de nodos que se han saltado, etc.

Este cuadro de información, visible en la ilustración 24, muestra también un listado de los métodos que están involucrados en la animación.

Por otro lado, al pulsar con el botón derecho sobre un nodo, se puede activar un cuadro de diálogo que da información sobre el nodo en cuestión: estado de ejecución, número de hijos, número de subnodos que dependen de él, método al que pertenece, etc. Se puede ver una muestra de este cuadro de diálogo en la ilustración 25.

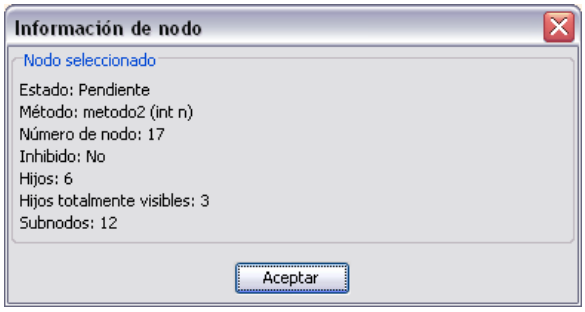

Ilustración 25 Cuadro de información de un nodo

4.2.8. Editor de código

SRec proporciona un editor básico de código Java. Este editor, cuya ventana se puede apreciar en la ilustración 26, ofrece la posibilidad de realizar modificaciones sobre la clase Java que se encuentra cargada en la aplicación o bien sobre cualquier otra que se elija desde el sistema de archivos.

Una vez se han realizado los cambios deseados, se puede llamar al compilador de Java directamente desde esa ventana para comprobar que los cambios introducidos no provocan errores de compilación.

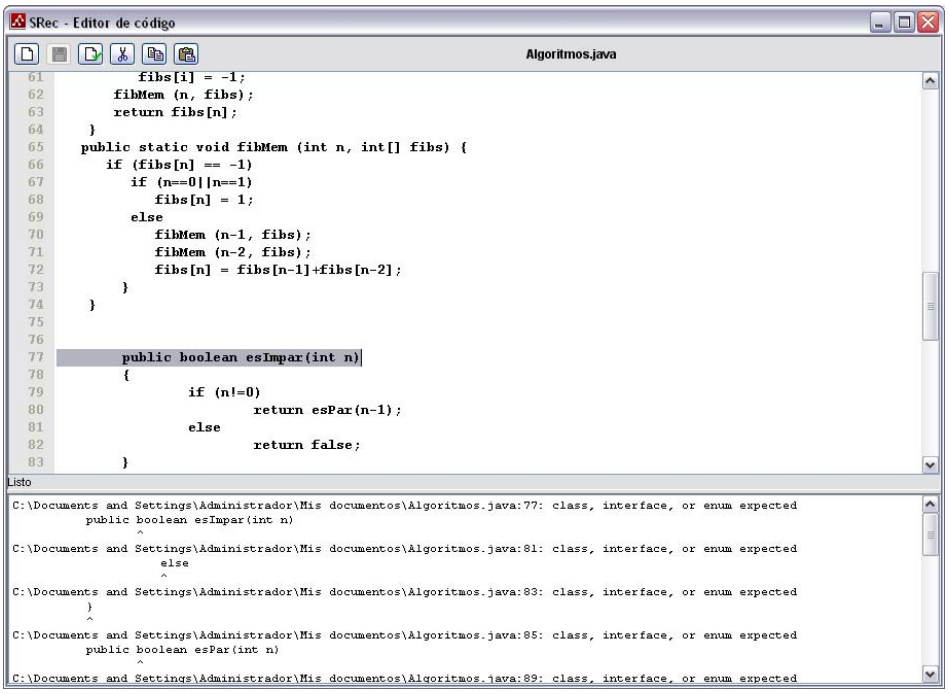

Ilustración 26 Ventana de edición de código

Si existen errores, estos aparecerán en el área inferior de la ventana con el mismo formato que ofrece el compilador de Java. Así, se podrá conocer la línea de código en la que se ha detectado el error, junto con una breve definición del mismo. Al seleccionar en el área inferior la línea de texto que indica la línea de código donde se ha localizado un error, ésta aparecerá señalada en el propio código (tal y como se ve en la ilustración).

El editor da opción de cortar, copiar y pegar texto, con las mismas posibilidades que otros editores de texto plano. Una vez que el usuario cierre el editor, podrá procesar la clase editada para que los cambios tengan efecto sobre las siguientes visualizaciones que se generen.

4.2.9. Gestión de configuración

SRec proporciona una configuración por defecto que el usuario puede personalizar completamente. Esta configuración por defecto es cargada cada vez que se inicia el programa. Así, una vez que el usuario haya seleccionado la configuración exacta de todos los parámetros, debe guardar la configuración actual del programa como configuración por defecto. Sólo así podrá hacer uso de ella cada vez que inicie la aplicación sin necesidad de reconfigurar el programa.

A medida que se va utilizando SRec durante la sesión, pueden ir variando los parámetros de configuración seleccionados, por lo que, para recuperar la configuración por defecto será necesario utilizar la opción de menú correspondiente a la restauración de la configuración.

SRec además permite almacenar y cargar distintas configuraciones diferentes, que pueden ser necesarias según el contexto en el que se quiera utilizar el programa. De esta manera, se pueden recuperar distintas configuraciones de una forma cómoda y totalmente adaptada al usuario sin necesidad de reconfigurar el programa cada vez.

El almacenaje de estas configuraciones se realiza en documentos XML, lenguaje empleado extensamente en diversos aspectos del programa.

4.2.10. Idioma

SRec puede ser utilizado en español y en inglés, aunque está diseñado de tal forma que resulta relativamente fácil añadir un nuevo idioma en el que el programa pueda funcionar.

Esto es debido a que todos los textos están almacenados en un documento XML que puede ser ampliado con otros idiomas siguiendo una serie de directrices que se puede deducir tras observar la estructura del propio documento XML.

SRec permite la selección del idioma a través de un sencillo cuadro de diálogo donde aparecen los idiomas disponibles y el que está actualmente seleccionado.

4.2.11. Exportación de visualizaciones

SRec da la posibilidad de exportar las visualizaciones en formato gráfico para que pueda utilizarse dicha salida en otras aplicaciones de todo tipo, tales como navegadores web (Internet Explorer, Mozilla Firefox, Opera…), aplicaciones de ofimática (procesadores de texto, hojas de cálculo, presentaciones electrónicas…), programas orientados al diseño gráfico (CorelDraw, Photoshop, Suite Macromedia…).

SRec puede exportar imágenes en formato PNG, GIF y JPG de cualquier estado de la visualización, o bien realizar una colección de imágenes, una por cada estado de la visualización. SRec también es capaz de generar animaciones GIF que, en un único archivo, muestran la secuencia completa de la animación. Éste puede ser cargado por SRec con posterioridad para poder visualizar la animación almacenada, aunque este tipo de formato no permite ningún tipo de interactividad respecto al manejo de la animación, el formato o la cantidad de información que se muestra.

Como opción adicional, SRec permite también exportar el contenido de la vista de traza en un documento estándar HTML, compatible con los navegadores web.

De esta forma, SRec permite la utilización de la web como ventana de difusión para la divulgación de algoritmos con carácter global y universal, al no necesitar de ningún programa (ni siquiera del propio SRec) para poder ver estas imágenes, animaciones GIF y documentos HTML. Todos estos archivos son visualizables en cualquier navegador web y las imágenes de formato estándar son fácilmente incrustables en documentos de todo tipo como PDF, creados con editores de texto como Microsoft Word o creados con LaTeX.

4.2.12. Sistema de ayuda

SRec proporciona un sistema de ayuda representado en la ilustración 27, que se basa en la navegación web para ofrecer de una manera ágil información sobre cómo aprovechar las funcionalidades que proporciona el programa. Aparece dividida en varios apartados y en cada una de sus páginas se puede navegar cómodamente hacia la siguiente y la anterior página de información relacionada.

Además, en todo momento se permite acceder al índice general de la ayuda, para que el usuario pueda llegar a la información que necesite en apenas un par de clics.

Ilustración 27 Ventana de ayuda

La ayuda está dividida en los siguientes puntos:

- Visión general de SRec: ofrece información general sobre la aplicación.
- Visualizaciones: proporciona abundante ayuda sobre la generación, carga, almacenamiento y utilización de las visualizaciones.
- Configuración de visualizaciones: aporta explicaciones sobre las posibilidades de personalización de las visualizaciones (formato, opciones de visión de datos...).
- Editor de código: referencia básica sobre el editor de archivos Java que incorpora SRec.
- Otras opciones: se documentan algunas otras opciones de configuración del programa.
- Más ayuda: enlaza a la web del Departamento responsable de este programa para que el usuario pueda solicitar más información si lo desea.

La ayuda, al igual que el resto del programa, se ofrece tanto en inglés como en español. El acceso a cada idioma es automático en función de la configuración seleccionada.

4.2.13. Gestión de errores de compilación

Al utilizar como entrada un archivo de código se corre el riesgo de que éste no sea correcto, por lo que en este caso SRec lanzará un cuadro de error como el de la ilustración 28 avisando de que no fue posible el procesado de esa clase al tener lugar uno o más errores de compilación que SRec detalla a continuación en el propio mensaje de error.

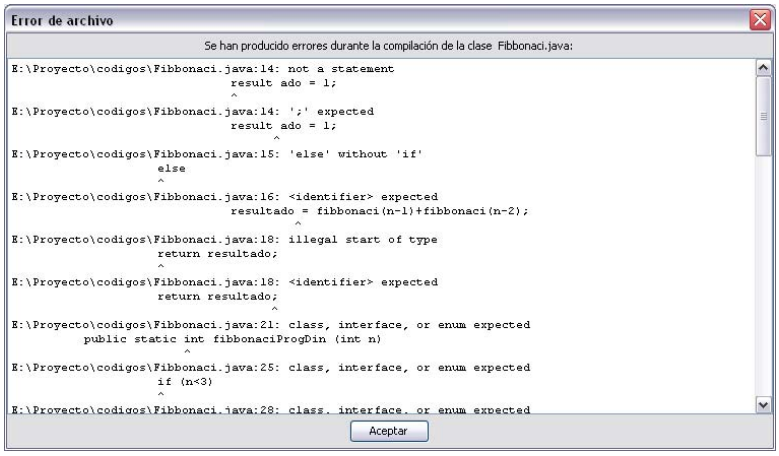

Ilustración 28 Cuadro de errores de compilación

Éste es uno de los momentos en que resulta más útil el editor básico de código que incorpora SRec.

4.2.14. Gestión de errores de ejecución

Durante la ejecución de los algoritmos pueden producirse errores de ejecución (recursividad infinita -provoca *overflow* en la pila-, navegación por un array fuera de los índices válidos...) que son recogidos por el programa. SRec muestra el correspondiente mensaje de error y deja la ventana vacía al no poderse llevar a cabo la visualización.

Ilustración 29 Cuadro de error

Para poder generar una visualización es necesario que no se produzcan errores de ningún tipo durante la compilación y ejecución del algoritmo.

4.3. Otras partes del programa

A continuación se especifican otras partes del programa no mencionadas hasta ahora junto a algunas características que pueden resultar de interés y aspectos de presentación de la aplicación.

4.3.1. Inicio de la aplicación

Al iniciarse, SRec carga las opciones de configuración por defecto que tiene almacenadas. Además, SRec muestra durante su carga una ventana de presentación al estilo de otros programas como la conocida *suite* de ofimática Microsoft Office (ver ilustración 30).

Ilustración 30 Imagen de presentación de la aplicación

Si SRec detecta que no hay configurada una dirección válida para encontrar la máquina virtual de Java lanza el cuadro de selección de máquina virtual para poder configurar correctamente el programa. No configurar la máquina virtual correctamente impedirá a la aplicación poder generar animaciones nuevas, aunque sí podrá cargar animaciones guardadas en sesiones anteriores en formato XML.

Una vez termina el proceso de arranque, los paneles de la ventana de la aplicación permanecen vacíos, a la espera de que el usuario decida generar o cargar una visualización.

4.3.2. Procesado de archivos

El procesado de los archivos Java sigue la siguiente secuencia de pasos: SRec se abre en un estado inicial en el que no tendrá ninguna clase preprocesada. El usuario podrá procesar una clase para que el programa determine qué métodos se pueden ejecutar para su visualización. Esta clase se procesará sólo una vez y se podrán lanzar animaciones con sus métodos tantas veces como desee el usuario.

Al inicio de este proceso, el usuario debe elegir la clase que desea utilizar. SRec analiza el código y prepara aquellos métodos que son visualizables (no se pueden visualizar métodos declarados en clases abstractas, métodos que no devuelvan valor primitivo alguno, métodos que no tengan parámetros o que alguno de ellos no sea primitivo o bien métodos que tengan parámetros de más de dos dimensiones).

Una vez que finaliza el procesamiento de la clase, ésta se mantiene almacenada de manera indefinida para que el usuario pueda iniciar tantas visualizaciones como desee sobre los métodos elegidos a lo largo de la sesión.

SRec informa de la existencia de errores en el código mediante cuadros de diálogo informativos. SRec mostrará en un cuadro los errores encontrados por el compilador de Java. El usuario podrá emplear el editor de código que proporciona SRec para editar la clase y corregir los errores encontrados. Tras ello, deberá repetir el procesado de la clase.

Cuando el usuario desee utilizar una segunda clase para visualizar alguno de sus métodos, tendrá que procesarla igualmente y a continuación podrá generar tantas visualizaciones como necesite sobre sus métodos. La primera clase que procesó queda eliminada, por lo que sus métodos dejarán de estar disponibles para nuevas visualizaciones.

4.3.3. Máquina virtual de Java

SRec necesita contar con el Java Development Kit (JDK) 6 para poder compilar los archivos de código Java que recibe a su entrada. SRec ofrece un cuadro de diálogo para que el usuario seleccione dónde se encuentra la versión adecuada del JDK con el fin de que SRec pueda compilar y ejecutar clases.

Es fácilmente modificable la versión de JDK que SRec utilizará para compilar y ejecutar los programas, ya que las diferentes versiones proporcionan diferentes rendimientos que pueden interesar según el algoritmo y la versión de Java para la que fue escrito.

4.3.4. Configuración de archivos intermedios que se generan

Durante el procesado de las clases se generan una serie de archivos intermedios que pueden ser interesantes de almacenar para el profesor o alumno con el fin de obtener, por ejemplo, una representación XML estándar del código o la compilación de la clase procesada. Por defecto no se mantienen ninguno de los archivos intermedios generados, aunque SRec ofrece la posibilidad de guardarlos en el directorio donde se encuentra el código original que se quiere procesar.

Los archivos intermedios que se pueden mantener en el sistema de ficheros del usuario son:

- Representación XML del código Java original: el código Java es representado en forma de documento XML estándar. Así se obtiene una representación estándar del código que puede resultar interesante para su estudio.
- Representación XML del código Java generado: SRec genera un código Java ampliado que permite almacenar los valores de la ejecución, el programa da la posibilidad de guardarlo en formato XML.
- Código Java generado: también se puede almacenar el código Java generado, que contiene líneas adicionales para gestionar el almacenamiento de la traza de la ejecución.
- Código Java original compilado: SRec compila los programas pasados a su entrada para comprobar que son correctos. El programa ofrece la posibilidad de guardar la clase compilada para que el usuario no necesite compilarla si desea utilizarla fuera de SRec.
- Código Java generado compilado: SRec también permite almacenar el código Java que realmente se ejecuta en SRec para poder mostrar la visualización. No obstante, el interés que pueda tener el usuario en almacenar este archivo es, *a priori*, muy bajo.

5. Diseño

A continuación se desarrollan algunas cuestiones de diseño referidas a los diversos elementos que forman la interfaz del programa (menús, cuadros de diálogo, barra de animación, etc.), a las labores de preprocesamiento de clases y a la arquitectura general de la aplicación.

5.1. Interfaz de usuario

La interfaz de usuario del programa está formada por las tres ventanas con que cuenta el programa (ventana principal, ventana de edición de código, ventana de ayuda), los diferentes cuadros de diálogo (cada tipo de ellos mantiene un objetivo diferente), los menús, la barra de herramientas y la barra de animación (estos tres últimos elementos se encuentran integrados en la ventana principal).

5.1.1. Ventana principal, menús y barra de herramientas

La ventana principal de SRec es la encargada de albergar las visualizaciones que genera el programa y de habilitar el acceso a todas las opciones que ofrece el programa para su configuración y gestión de su funcionalidad. Así, la ventana queda dividida en tres partes principales: menú, barra de herramientas y área de visualización, que ocupa una amplia mayoría del área de la ventana.

El menú de SRec está compuesto por tres categorías. La primera está orientada a las acciones de carga y gestión de archivos, la segunda a la configuración del programa y la tercera y última a la ayuda del programa:

Archivo: permite realizar todas las tareas referidas a la gestión de clases (preprocesado, edición) y visualizaciones (carga, generación, guardado, exportación).

- Editar clase…: Abre el cuadro de diálogo que permite seleccionar un archivo de código Java para poder editarlo después mediante la ventana de edición de código.

- Editar clase procesada: Requiere que exista una clase procesada. Abre la ventana de edición de código con el archivo Java perteneciente a la clase que actualmente tiene preprocesada el programa.
- Cargar y procesar clase…: Abre el cuadro de diálogo que permite seleccionar un archivo de código Java para procesarlo y preparar el listado de métodos visualizables. Es el paso necesario para iniciar el preprocesamento de una clase.
- Reprocesar clase: Requiere que exista una clase procesada. Reprocesa la misma clase que ya está procesada. Puede tener sentido si se han introducido cambios en el código o si se desea visualizar un método que no fue procesado anteriormente.
- Nueva animación: Requiere que exista una clase procesada. Muestra el listado de métodos que fueron procesados en la clase que actualmente se encuentra disponible en la aplicación. Una vez que el usuario escoja el método e introduzca los parámetros necesarios, podrá utilizar la visualización.
- Cargar animación: Abre el cuadro de diálogo que permite seleccionar un documento XML que contenga una visualización generada por SRec con anterioridad. Tras la carga y lectura de la información, aparecerá la visualización en la ventana del programa.
- Cargar animación GIF: Abre el cuadro de diálogo que permite seleccionar un archivo gráfico GIF que contenga una animación secuencial generada por SRec con anterioridad. Tras la carga, aparecerá la animación en la ventana del programa, aunque no proporcionará ningún tipo de interacción debido a las restricciones del propio formato estándar GIF.
- Guardar animación: Requiere que exista una visualización abierta. Guarda en un documento XML la visualización actual. Abre el cuadro de diálogo que permite definir un documento XML de destino para la visualización actual. Tras ello, la visualización se guarda en disco.
- Guardar traza: Requiere que exista una visualización abierta. Guarda en un documento HTML la información contenida en ese instante en la vista de traza. Abre el cuadro de diálogo que permite definir un documento HTML de destino para la vista de traza actual. Tras ello, la información se guarda en disco en el citado formato estándar.
- Exportar animación como GIF: Requiere que exista una visualización abierta. Tras abrir el cuadro de diálogo para seleccionar el archivo GIF de destino, genera una animación compuesta por toda la secuencia completa de estados de la visualización que es guardada en formato GIF animado.
- Exportar capturas de animación: Requiere que exista una visualización abierta. Tras abrir el cuadro de diálogo para seleccionar el nombre genérico de los

archivos PNG, GIF o JPG que se generarán, genera una animación compuesta por toda la secuencia completa de estados de la visualización, guardando cada estado en un archivo diferente.

- Exportar captura: Requiere que exista una visualización abierta. Tras abrir el cuadro de diálogo para seleccionar el archivo GIF de destino, guarda en él una captura del estado actual de la visualización.
- Salir: Cierra el programa de manera incondicional.

Configuración: Este menú permite adaptar SRec a las necesidades que el usuario tenga en cada momento.

- Información de la animación: Requiere que exista una visualización abierta. Gracias a un cuadro de diálogo se muestran algunos datos de interés sobre la animación, como por ejemplo, número total de nodos, número de nodos visibles, número de nodos históricos, número de nodos que están ocultos por el usuario...
- Búsqueda de llamadas: Requiere que exista una visualización abierta. Permite mediante el cuadro de diálogo que abre realizar una búsqueda de llamadas sobre todas las que componen la ejecución del algoritmo que se está visualizando. Los criterios se basan en el método escogido y los valores de los parámetros introducidos.
- Visibilidad de métodos y parámetros: Requiere que exista una visualización abierta. Permite definir mediante un cuadro de diálogo la visibilidad de cada uno de los parámetros de todos los métodos implicados en la visualización. De esta manera, el programa permite elegir qué datos se ven y cuáles se omiten en cada momento de la visualización, por lo que la flexibilidad en el uso de la herramienta crece de manera cualitativa en gran medida.
- Formato de árboles..: Permite definir mediante un único cuadro de dialogo qué datos se mostrarán en el árbol de activación (todos los datos, sólo valores de parámetros de entrada, sólo valores de retorno), qué se hace con los nodos históricos del árbol (mantener, atenuar, suprimir), si se muestran o no los árboles pertenecientes a llamadas sobre las cuales se aplica un salto, y si debe aparecer en la parte inferior de la vista el visor para ayudar en la navegación por árboles muy grandes.
- Formato tipográfico…: Da la posibilidad mediante un cuadro de diálogo dividido en tres pestañas de configurar los colores que se utilizan en todas las vistas de la visualización. Además, da la opción de cambiar los tipos de letra de algunas vistas y las clases de flechas empleadas. También deja escoger los bordes estilizados que se desea aplicar a las celdas o las distancias que hay entre las

mismas. El grosor de las flechas o el de los marcos son otros de los parámetros configurables.

- Configuración de zoom…: Requiere que exista una visualización abierta. Permite configurar el efecto de zoom sobre las vistas del árbol de activación y de la pila de control. Da la opción de introducir un valor numérico manualmente, o bien de realizar ajustes graduales. Adicionalmente se puede ajustar el grafo al tamaño del panel de visualización.
- Archivos intermedios…: Durante el preprocesado de clases se generan una serie de archivos intermedios que pueden ser de interés para el usuario. Mediante un cuadro de diálogo se puede configurar qué ficheros se deben mantener y cuáles se deben eliminar.
- Máquina Virtual de Java…: Se activa automáticamente si la configuración no es correcta. Mediante un cuadro de diálogo permite buscar y seleccionar una Máquina Virtual de Java válida que permita a SRec compilar clases Java.
- Idioma…: Permite seleccionar entre alguno de los idiomas disponibles mediante un sencillo cuadro de diálogo.
- Mostrar/ocultar barra de herramientas: Hace aparecer y desaparecer la barra de herramientas.
- Restaurar configuración: Carga la configuración por defecto del programa.
- Abrir configuración…: Abre el cuadro de diálogo que permite seleccionar un documento XML que contenga una configuración válida para SRec. Tras la selección carga los valores para los diferentes parámetros, adquiriendo la configuración definida,
- Guardar configuración…: Abre el cuadro de diálogo que permite seleccionar un documento XML de destino que contendrá la configuración actual del programa. Tras ello, el programa almacena la configuración para que pueda ser recuperada con posterioridad.
- Guardar configuración por defecto: Guarda la configuración actual como configuración por defecto del programa.

Ayuda: Contiene las opciones que suelen ser típicas de este menú en la mayoría de programas acerca del manejo de SRec y de la identificación del mismo.

- Temas de ayuda…: Abre el visor de la ayuda, una ventana basada en la navegación por hiperenlaces donde se puede encontrar la ayuda básica del programa.
- Sobre SRec: Ofrece una información básica para identificar a SRec y al Departamento de la Universidad Rey Juan Carlos responsable de su creación.

Por otra parte, SRec cuenta con una barra de herramientas (ver ilustración 31) que permite acceder rápida y cómodamente a las opciones más importantes y utilizadas proporcionadas a través de menú. Esta barra puede deshabilitarse a través del menú Configuración, presentado anteriormente.

Ilustración 31 Barra de herramientas

Presenta cinco grupos diferenciados de botones: uno está orientado a la edición y procesado de clases, otro a la creación, carga y almacenamiento de visualizaciones, un tercero que permite hacer uso de opciones sobre la animación abierta, un cuarto centrado en la configuración de las visualizaciones (formato, opciones, zoom) y un quinto que proporciona el acceso directo a la ayuda del programa.

Los iconos que contiene la barra de herramientas se corresponden con los presentados en los menús, ya que todas las funcionalidades a las que da acceso rápido la barra están contenidas en el menú de la aplicación.

La tercera y última parte es la más importante, ya que se centra en el panel que contiene las visualizaciones cuando éstas están activas. En éste área se habilitan varias subáreas (sólo visibles cuando la visualización está activa) con distintas funcionalidades que se detallan a continuación, complementado con una ilustración:

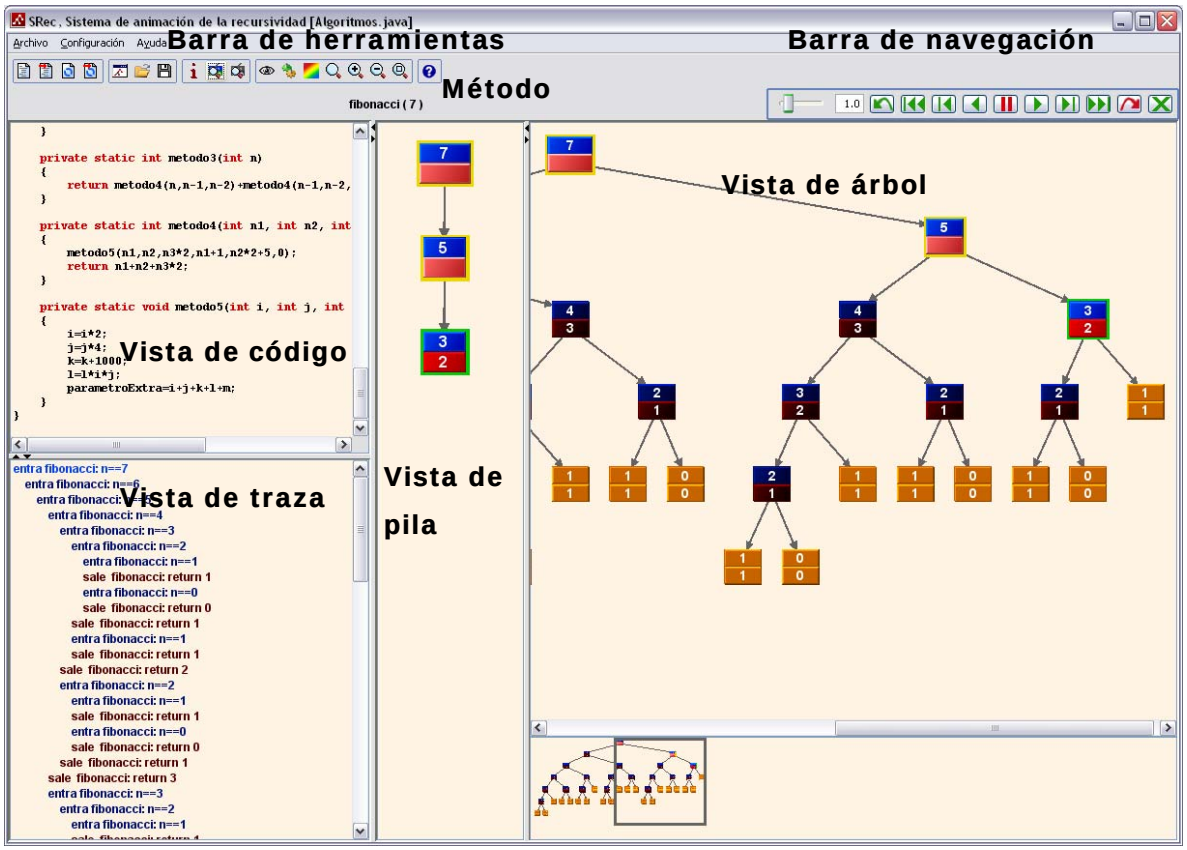

Ilustración 32 Vista general de SRec

- Área de título: Refleja el nombre del método principal que se está ejecutando, junto con los parámetros iniciales que se introdujeron por parte del usuario. Así se permite tener constancia en todo momento del algoritmo que se está visualizando.
- Área de control: Incluye una serie de botones y herramientas que permiten controlar el estado de la visualización, ofreciendo la posibilidad de realizar saltos hacia atrás, ir al principio de la visualización, dar un paso hacia atrás, iniciar la animación hacia atrás, pausar la animación, iniciar la animación hacia delante, dar un paso hacia delante, avanzar la visualización hasta el final y cerrar la visualización.
- Vista del árbol de activación: es la principal de las cuatro vistas, ya que en ella se representa el árbol de activación, que contiene toda la información sobre la ejecución del algoritmo que se está visualizando. En ella se puede ir viendo la transición entre estados según se maneje la visualización a través del área de control.
- Vista de la pila de control: ofrece información sobre las llamadas que se encuentran almacenadas en la pila. Su formato va íntimamente ligado al del árbol de activación.

- Vista de traza: representa textualmente todas las llamadas realizadas, indicando claramente sus parámetros, y siempre de manera secuencial y debidamente tabulada para una más fácil comprensión. También aparecen líneas de salida, que representan las finalizaciones de las subllamadas, donde se expresan los valores de retorno calculados.
- Vista de código: permite visualizar en la misma ventana que contiene la visualización el código de la clase Java que contiene el método que se está ejecutando. Así, se ayuda a la comprensión del algoritmo que se está visualizando, ya que en una misma ventana se tiene el código que se ha ejecutado y su resultado.

5.1.2. Cuadros de diálogo

SRec cuenta con varios tipos de cuadros de diálogo que permiten interactuar con el usuario en diferentes escenarios y con diferentes objetivos. Se listan y explican a continuación tales elementos de la interfaz del programa.

Cuadro "Acerca de": Ofrece la información identificativa de SRec, el Departamento de Lenguajes y Sistemas Informáticos I y de la Universidad Rey Juan Carlos de Madrid.

Cuadro de error: Este cuadro aparece cuando surge un error de ejecución de algún algoritmo, cuando hay algún problema al cargar un documento XML (visualizaciones, configuraciones, valores para parámetros...) o cuando se da un error durante la visualización.

Cuadro de error de compilación: Cuadro que aparece cuando SRec detecta que el compilador de Java devuelve errores por su salida estándar. SRec recoge estos errores y los muestra dentro de un cuadro de error diseñado apropiadamente en cuanto a tamaño y forma.

Cuadro de parámetros: Este cuadro permite la inserción de valores para los parámetros de los que consta el método que se pretende ejecutar para realizar posteriormente su visualización. Permite seleccionar qué parámetros aparecerán en la vista y cuáles quedarán ocultos.

Cuadro para generar valores aleatorios: Este cuadro, mostrado en la ilustración 33, es la interfaz que utiliza SRec con el fin de permitir al usuario la generación de valores aleatorios para los parámetros de un algoritmo que se desea ejecutar. Permite seleccionar para qué parámetros se desean calcular valores aleatorios así como introducir rangos de valores. Además, da la opción de generar valores ordenados o desordenados para arrays y matrices. El usuario puede introducir las dimensiones de éstos o bien darles un tamaño igualmente aleatorio.

Ilustración 33 Cuadro de generación de parámetros

Cuadro de métodos disponibles: Este cuadro permite al usuario elegir un método entre los que se encuentran en la lista, que son aquellos que fueron procesados anteriormente por el programa cuando cargó la clase elegida por el usuario. Este cuadro muestra, además del nombre de los métodos, los tipos de sus argumentos.

Cuadro de opción sobre el borrado de archivos intermedios: Este cuadro (ilustración 34) permite gestionar qué archivos intermedios se desean generar durante el procesado de clases para su almacenamiento permanente en el sistema de ficheros (misma carpeta a la que pertenece la clase que se procesa).

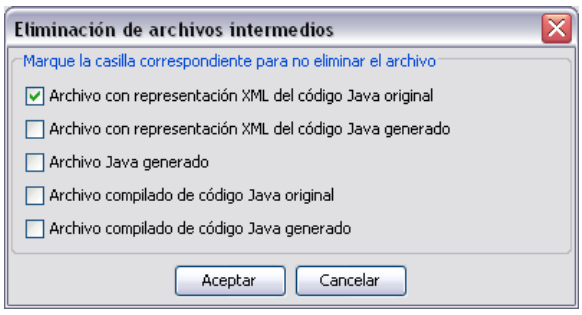

Ilustración 34 Cuadro de generación de archivos

Cuadro de opción de configuración de la visualización: Este cuadro permite, mediante sus tres pestañas, configurar el formato (colores, distancias, tipos de flechas, biselados de celdas, tipos de fuente...) de las visualizaciones (vistas de árbol de activación, pila de control, traza y código).

Cuadro de opción de la Máquina Virtual de Java: Este cuadro permite seleccionar qué máquina virtual se desea que emplee SRec para la compilación de clases y el ejecución de las mismas.

Cuadro de opción de configuración de las visualizaciones: Permite realizar ciertos cambios sobre las visualizaciones, referidos a la cantidad de información que se muestra y en qué condiciones.

Cuadro de idioma: Permite al usuario elegir en qué idioma desea utilizar el programa entre los idiomas disponibles. El cuadro muestra el idioma actualmente escogido y el listado de idiomas que se pueden emplear. Una vez que se elija un nuevo idioma, la ventana se reinicia para acoger los textos del nuevo idioma.

Cuadro de información: Es muy parecido al cuadro de error, sólo que está destinado a la notificación de mensajes de información general (notificación de exportaciones correctas, procesos finalizados adecuadamente...), no de errores surgidos durante la ejecución de alguna tarea.

Cuadro de introducción: Contiene la imagen que aparece durante la carga de SRec, que se mantiene en pantalla unos pocos segundos. Después desaparece, no permite ningún tipo de interacción.

Cuadro de progreso: Contiene, como se puede ver en la ilustración 35, una barra de progreso que informa, junto con información textual, de qué grado de completitud lleva alcanzado el actual proceso en ejecución. No permite ser cerrado por el usuario.

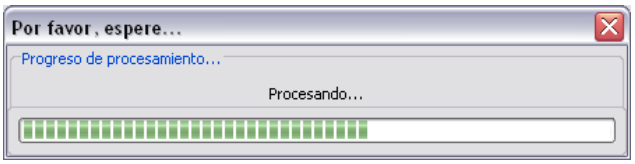

Ilustración 35 Cuadro de progreso

Cuadro de pregunta: Este cuadro plantea preguntas al usuario que puede contestar positiva o negativamente. De su respuesta dependerá el posterior comportamiento del programa.

Cuadro de pregunta sobre nueva visualización: Este cuadro está diseñado para plantear al usuario la necesidad de deshacerse de la visualización actual antes de iniciar otra.

Cuadro de Zoom: Permite gestionar el zoom de las visualizaciones sobre la vista del árbol de activación y de la vista de la pila de control. Permite el encaje de los grafos así como aumentos y disminuciones de zoom graduales. También ofrece la posibilidad de insertar el valor numérico deseado de manera directa.

5.2. Estudio de alternativas

Una de las decisiones más importantes a la hora de diseñar el programa que se deseaba realizar, fue determinar qué alternativa escoger respecto a si se deseaba animar algoritmos o bien visualizar programas.

La primera opción dotaría a SRec de la capacidad de generar animaciones de carácter abstracto para algoritmos concretos de manera independiente a códigos y lenguajes, lo que facilitaría la generación de las mismas, mientras que la segunda se basa en la ejecución de un código concreto, aportado por el usuario. Esto abre la puerta a un amplio abanico de algoritmos, tantos como el usuario desee programar en el lenguaje indicado (en este caso, Java).

Teniendo en cuenta que la intención del programa es facilitar la tarea de generación de visualizaciones, se apostó por la segunda opción, siendo conscientes de que sería necesario implementar un complejo proceso de análisis de las clases para lograr una gran automatización en la generación de las visualizaciones.

Además, surgieron dos alternativas referidas al manejo del código del algoritmo para poder extraer la información necesaria sobre la ejecución del mismo:

- Modificar el procesador del lenguaje, sin modificar el código fuente: esta alternativa fue desechada por su alta complejidad, ya que suponía la implementación de todo un sistema creado *ad-hoc* capaz de leer y analizar el código que permitiese obtener un esquema de su ejecución.

- No modificar el procesador del lenguaje sino el código fuente: esta posibilidad resulta totalmente factible al realizar conversiones a formato XML del código original, resultando así muy sencilla su manipulación y alteración utilizando las tecnologías DOM y SAX. De esta forma, se permite la inclusión de sentencias adicionales que permitan extraer de la ejecución la información necesaria para la creación de una traza que se utilizará como base de información a la hora de generar la visualización.

5.3. Preprocesamiento de clases

SRec proporciona una realización de visualizaciones de programas totalmente automatizada ya que se encarga de analizar y procesar automáticamente las clases seleccionadas por el usuario para que el coste de generación sea mínimo. Es por ello que SRec hace frente a un complejo proceso de análisis y modificación de las clases Java que el usuario le aporta a la entrada. Éste aparece esquematizado en la ilustración 36.

Como ya se ha hecho notar, SRec toma a su entrada un archivo de código Java que contiene declarada una clase en cuyo interior se encuentran los métodos que se desean visualizar. El primer paso que realiza SRec consiste en compilar la clase para comprobar que es léxica, sintáctica y semánticamente correcta, evitando así manipular inútilmente clases que ofrecen errores de compilación.

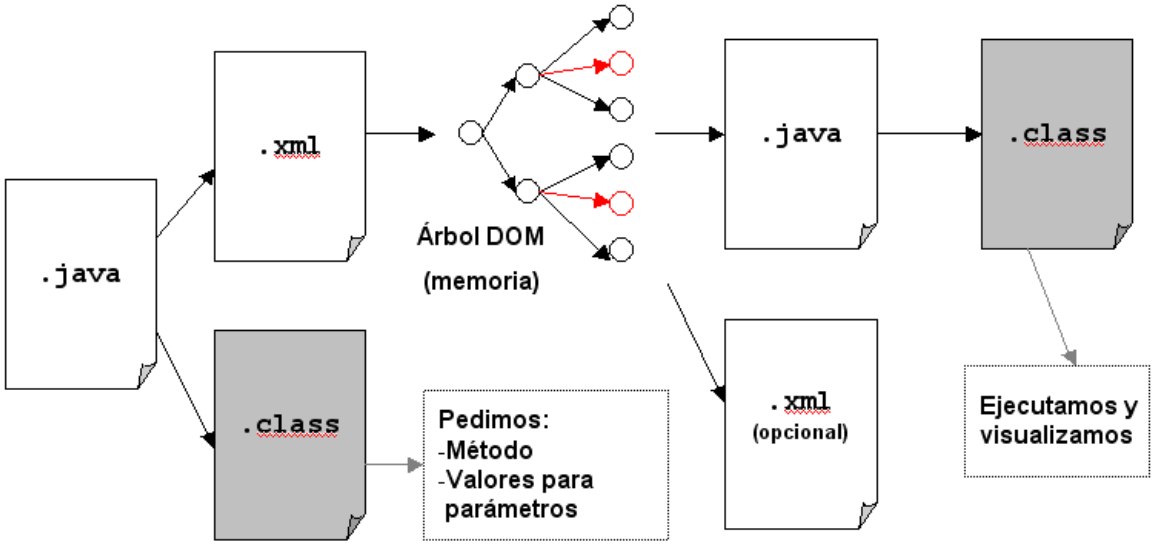

Ilustración 36 Esquema del preprocesado de clases

El siguiente paso consiste en convertir tal clase Java en un documento XML haciendo uso del paquete ToXML [16]. Este documento representa cada uno de los

elementos de los que consta la clase Java: directivas de importación de paquetes, pertenencia a paquetes, declaraciones de clases, métodos, atributos, variables locales... Además, también se generan nodos para estructuras de control (if-else, while, for, switch...), bloques, y todo tipo de sentencias (llamadas a métodos, asignaciones, operaciones...).

El documento XML ofrece así un árbol que en su primer nivel representa el archivo Java, en su segundo nivel alberga las sentencias de paquete y de importación, así como la declaración de clases. Este documento XML, de carácter temporal, puede ser salvado de manera permanente en el directorio de la clase original si así lo desea el usuario.

Dentro de los nodos correspondientes a la declaración de alguna clase, se pueden encontrar más niveles. El siguiente nivel contiene los atributos de instancia y de la clase, junto con los nodos de declaración de los métodos.

Dentro de los métodos, y viajando a niveles más profundos, se pueden encontrar nodos pertenecientes a las declaraciones de variables locales, a sentencias (simples o en bloque) y a estructuras de control. Igualmente, las estructuras de control y los bloques anónimos de sentencias implican la creación de más niveles de profundidad.

Por tanto, se obtiene una correspondencia unívoca, explícita y totalmente fiel al código Java original, lo que permitirá poder restaurar este código para su ejecución debidamente adaptado con posterioridad dentro de este proceso.

Esta conversión realizada tiene como finalidad facilitar la tarea de lectura y manipulación de los contenidos de la clase Java, al emplear la API de DOM que implementa Java para la gestión y manipulación de información contenida en documentos XML. Una vez que SRec ya dispone de una representación en lenguaje XML del código Java original, la aplicación revisa cuáles de los métodos que alberga la clase son aptos para ser visualizados, realiza una serie de adaptaciones sobre la clase y la mantiene posteriormente cargada. Entre las características que deben tener los métodos de la citada clase para que puedan ser visualizados se encuentran la obligatoriedad de que tengan al menos un parámetro de entrada y que no tengan parámetros o valores de retorno no primitivos.

Durante el procesamiento de la clase la aplicación inserta líneas de código adicional de una manera muy sencilla gracias a la manipulación con DOM sobre el árbol cargado en memoria que representa el documento XML al que se hacía referencia anteriormente. Las líneas de código, visibles en la ilustración 37, tienen como finalidad recoger los valores de los parámetros de entrada así como el de retorno, facilitando además su almacenamiento por parte de la aplicación en una traza interna que representa fielmente el árbol de llamadas que se genera en las llamadas recursivas que tienen lugar durante la ejecución del algoritmo.

```
public static int fibonacci ( int n )
    {
        Object pppppp01[] = new Object[1];
        pppppp01[0]=n;
        Traza.singleton().anadirEntrada(new Estado(pppppp01));
        int resultado = 0;
        if (n==1)
            resultado = 1;
        else
            if (n==2)
               resultado = 1;
            else
                resultado = fibonacci(n-1)+fibonacci(n-2);
        int zzzzzz01 = resultado;
        Object rrrrrr01[] = new Object[1];
        rrrrrr01[0]=zzzzzz01;
        Traza.singleton().anadirSalida(new Estado(rrrrrr01));
        return zzzzzz01;
    }
```

Ilustración 37 Código generado por SRec para su ejecución

Tras esto, se consigue tener un árbol que representa una clase Java totalmente preparada para la visualización de algoritmos en la aplicación con todas las garantías técnicas (ejecución, almacenamiento de información...) y satisfaciendo la demanda del usuario respecto a los métodos que se podrán visualizar tantas veces como desee. Esta clase Java puede ser salvada como fichero intermedio si así configura el usuario el programa.

El siguiente paso es volcar ese árbol al sistema de ficheros en forma de clase Java, por lo que se hace uso del traductor que se ha programado de carácter genérico para tal fin. De esta manera, el único paso que queda por realizar ya es compilar la clase recién generada, ya que será la que finalmente se ejecute a petición del usuario cuando desee generar una visualización.

5.4. Motor de las animaciones

Las visualizaciones cuentan con un motor lógico que es capaz de determinar, bajo varias circunstancias condicionadas por la configuración elegida por el usuario, cuál es el estado al que se debe acceder tras la intervención del usuario.

Por un lado, el motor determina cuál es el siguiente paso en la evolución de la visualización. En un primer momento, un nodo puede estar activo tras abrirse su llamada, por lo que en la visualización aparecerán los valores de sus parámetros pero no el valor de retorno de tal llamada.

A continuación, se activa la primera de las llamadas recursivas que contiene, quedando en el mismo paso en el que estaba su nodo padre un instante antes. Si este nodo no cuenta con nodos hijos (caso base), el siguiente paso es resolver tal llamada, por lo que su valor de retorno se hace visible.

El siguiente paso será la activación de la siguiente llamada recursiva, si es que la hay, del nodo padre, que pasará a mostrar únicamente los valores de sus parámetros de entrada. Así, se repite el proceso con este otro nodo de igual manera.

Finalmente, cuando todas sus llamadas recursivas han sido finalizadas, el siguiente estado se traducirá en la visualización de su valor de retorno, que será el valor final de la ejecución del algoritmo.

A continuación se ofrece en la ilustración 38 la secuencia de estados que tiene el ejemplo descrito empleando el algoritmo de los números de Fibonacci, introduciendo como valor de entrada “3”.

Este motor también se encarga de realizar transiciones entre distintos tipos de configuración, garantizando en todo momento la coherencia de los estados alcanzados y la correspondencia unívoca entre los estados de uno y otro tipo de configuración.

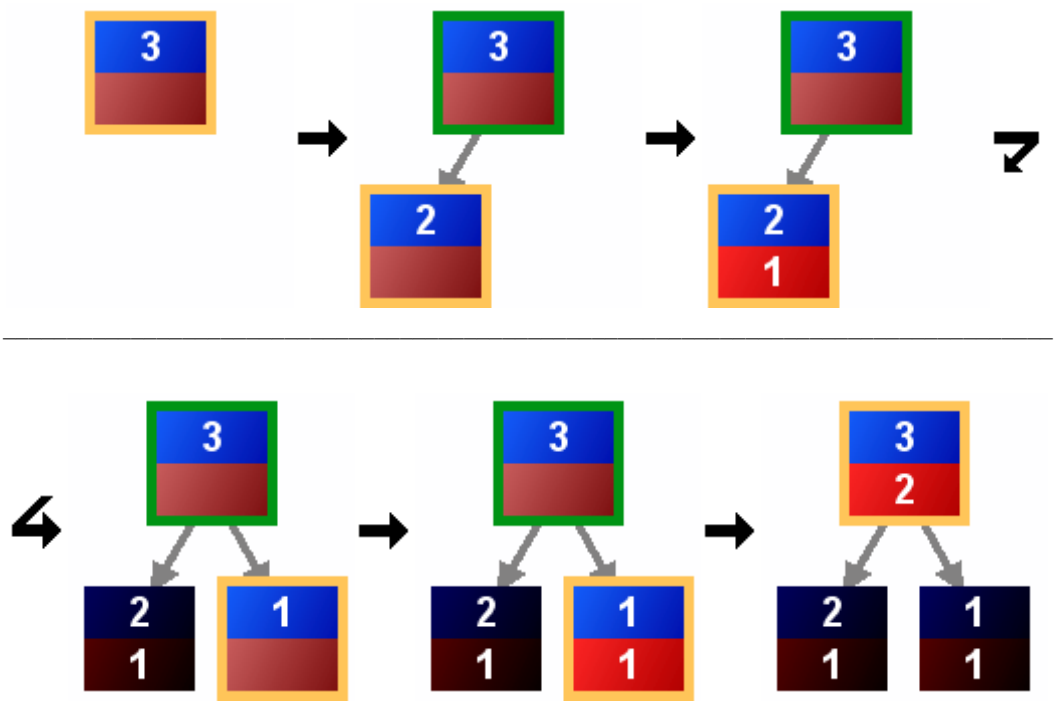

Ilustración 38 Secuencia de avance de una visualización mostrando todos los datos

Los principales cambios que soporta el motor atañen a la cantidad de información que se muestra en pantalla, ya que es capaz de gestionar procesos de visualización en los que sólo se muestran valores de parámetros de entrada o valores de retorno de las distintas llamadas recursivas.

Estos cambios en la configuración producen que el número de estados por el que debe pasar la visualización varíe para adaptarse a las características da cada tipo de configuración. Se presenta a continuación mediante la ilustración 39 la secuencia de estados para el mismo algoritmo visto anteriormente, pero mostrando sólo los valores del parámetro de entrada. Nótese que el número total de estados pasa de seis a cuatro.

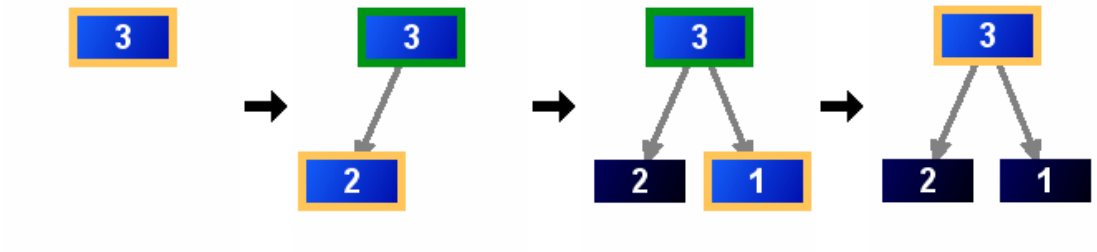

Ilustración 39 Secuencia de avance de una visualización mostrando sólo datos de entrada

5.5. Arquitectura de la aplicación

El diseño de SRec refleja la intencionalidad de que su arquitectura de clases sea lo más flexible y genérica posible para poder añadir diferentes tipos de visualizaciones en el futuro y favorecer así la implementación de nuevas herramientas que trabajen con otras técnicas de diseño, reaprovechando el trabajo y esfuerzo invertido en el desarrollo de SRec.

Así, el diseño arquitectónico se ha basado en el patrón arquitectónico de Modelo Vista Controlador. Este patrón diferencia tres partes en la arquitectura del programa. El modelo representa los datos y su lógica de control; en el caso de SRec los datos son la traza de ejecución de los distintos algoritmos que se visualizan, manejados por el motor lógico que ayuda a transitar a los diferentes estados de la visualización.

La vista se centra en la interfaz del programa que, como ya se ha explicado, está repartida en tres ventanas junto con todos sus elementos (menús, barra de animación, barra de herramientas...) y varios tipos de cuadros de diálogo. Ésta parte es igualmente fundamental, ya que en un programa de visualización la interfaz es crítica a la hora de valorar si el programa realmente cumple su objetivo.

Por último, el controlador se encarga de la interacción con el usuario. Maneja las situaciones que se dan aplicando sobre los datos (la traza) las acciones pertinentes tras dicha interacción.

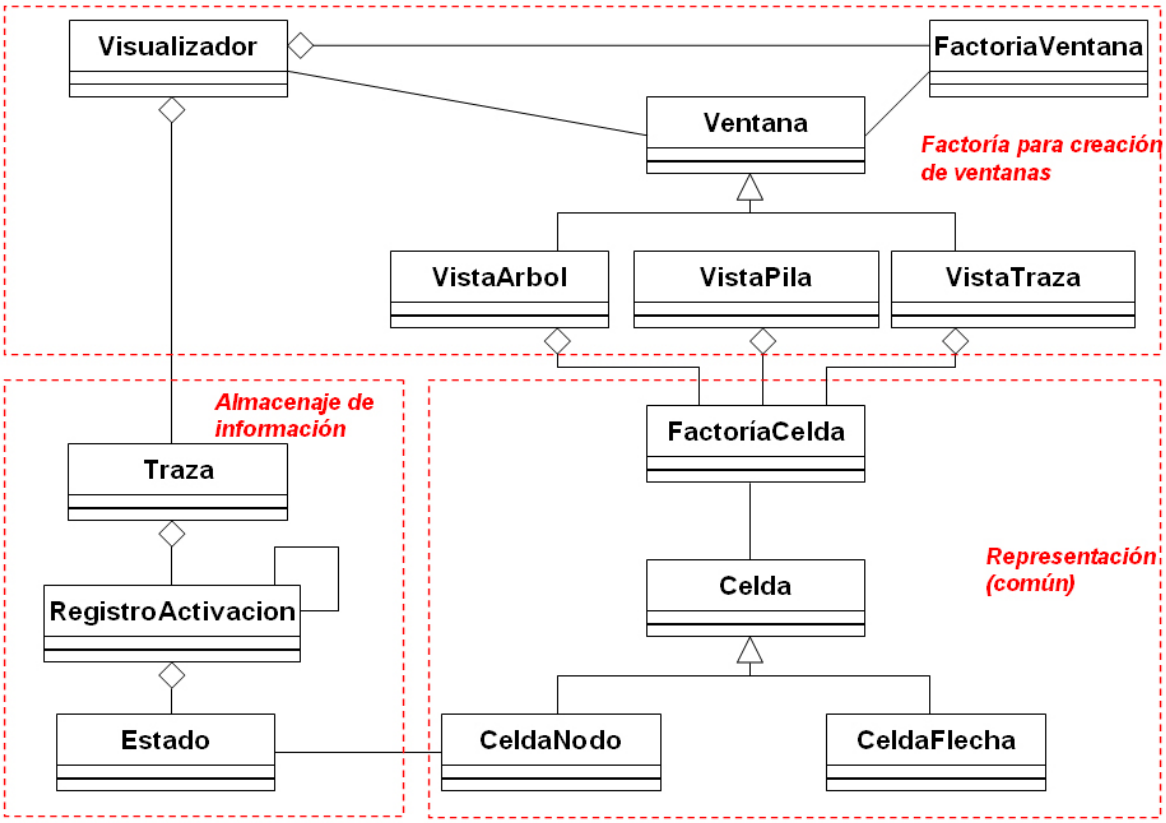

Ilustración 40 Arquitectura básica del programa

Como se puede ver en la ilustración 40, sólo se maneja una única copia de los datos y los elementos para dibujar los grafos de las diferentes vistas gráficas son los mismos. Lo único que cambia son las vistas de la visualización y la lógica de generación de las mismas.

Gracias a esta arquitectura, resulta extremadamente sencillo añadir nuevas visualizaciones. Es por ello por lo que se pueden ofrecer varias vistas simultáneamente en la ventana, y resultaría técnicamente viable dotar a SRec de la capacidad de alternar con otras posibles vistas si fuera necesario. Esta característica se traduce en una alta extensibilidad de la aplicación.

5.6. Estados de SRec

SRec puede encontrarse en múltiples estados que se detallan a continuación. Cuando finaliza la carga inicial de la aplicación, SRec queda en un estado de espera (“Espera 1” en el diagrama presentado en la ilustración 41), que permite al usuario cargar una

visualización existente o bien procesar una clase para la posterior visualización de la ejecución de alguno de sus métodos.

Al cargar una visualización se pasa al estado de visualización (“Visualización 1” en el diagrama), que permite realizar un uso totalmente funcional de la visualización cargada desde el sistema de ficheros.

Sin embargo, la opción que no está aún disponible en los estados comentados es la de poder generar una nueva visualización, ya que no ha sido procesada ninguna clase aún en la sesión actual de la aplicación.

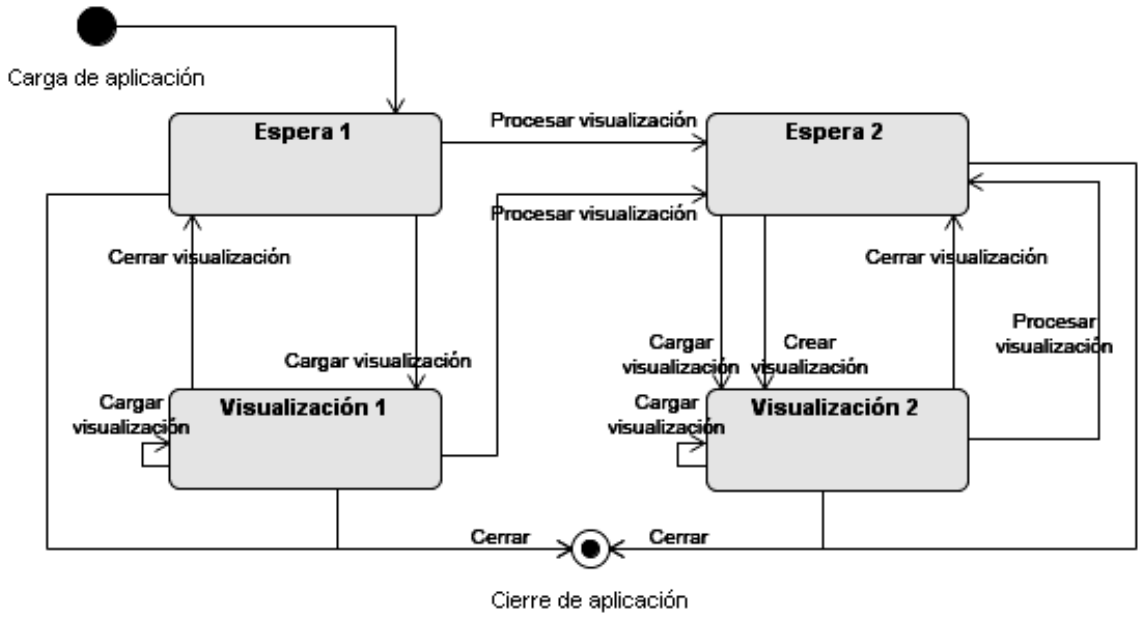

Ilustración 41 Diagrama básico de estados del programa, no se representan estados de error

Si en el estado de espera decidimos procesar una clase, pasaremos a otro estado de espera (“Espera 2” en el diagrama) que permitirá, aparte de procesar una nueva clase o de cargar una visualización existente, generar una nueva visualización desde la ejecución de un método perteneciente a la clase procesada.

Desde ese momento, siempre habrá una clase cargada tras su procesamiento, por lo que la posibilidad de generar nuevas visualizaciones siempre estará ya disponible mientras no finalice la sesión. La transición entre estados seguirá siendo entre estado de espera (“Espera 2”) o estado de visualización (“Visualización 2”).

Independientemente del estado en que se encuentre la aplicación, ésta puede ser cerrada instantáneamente pulsando la opción correspondiente de menú o bien con el botón de la ventana.

En algunos momentos, como por ejemplo al cargar o generar una visualización, se pueden producir errores (documentos corruptos, clases con errores de compilación, desbordamiento de pila por generaciones de visualizaciones demasiado extensas…) que no quedan recogidos en el diagrama presentado en la página anterior. Estos errores provocan que el programa quede en el estado anterior en el que se encontraba antes de producirse tal error.

6. Implementación

La división del código en paquetes, la importación y exportación de información, la integración de librerías externas o el desarrollo del sitio web de SRec son algunos de los temas que se tratarán a continuación.

6.1. Paquetes

Como ya se ha comentado, 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.

ToXML ***: 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, está programado por Harsh Jain, que 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 tres 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 otro lado, también se encuentra la ventana de edición de código, que permite, además de editar un archivo de texto, compilar clases Java para comprobar su corrección. 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 (como por ejemplo, PanelCodigo, PanelArbol, PanelPila o PanelTraza) o proporcionar 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 (PanelVentana, PanelContenedor) o bien ayudan a configurar la disposición de las vistas (PanelAlgoritmo) o el manejo de la interfaz de las mismas (PanelBotonesVisualizacion).

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: recoge 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 proyecto, 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 recibidas.

A continuación se presenta un diagrama de clases que relaciona todos los paquetes:

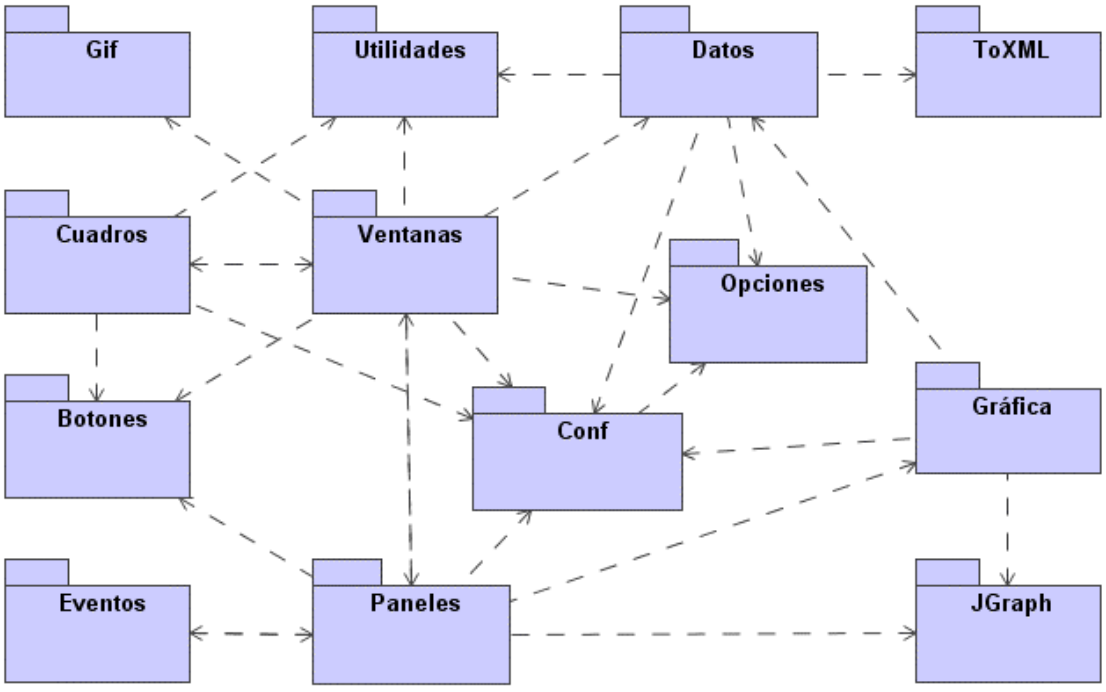

Ilustración 42 Diagrama de paquetes de SRec

6.2. Documentos XML generados

SRec emplea el lenguaje XML como formato de exportación e importación de información para su funcionamiento y el almacenamiento de visualizaciones. En concreto, SRec hace uso de documentos XML en los siguientes casos: textos de programa, almacenamiento de visualizaciones, almacenamiento de opciones de configuración, almacenamiento de parámetros para un algoritmo y representaciones de códigos Java.

6.2.1. 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 en realidad 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 cierta parametrización adicional para poder reproducir la visualización en las mismas condiciones.

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 exacto del documento XML pudieran cargar la traza de manera completa e incluso obviando el resto de información contenida en el documento XML.

Por otro lado, el documento XML contiene amplia información sobre el formato en que debe presentarse la traza. Esta información de formato satisface la parametrización que emplea SRec en sus visualizaciones, y que 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...

Además, también se almacena información adicional de formato, como el tipo de flechas que desea utilizar el usuario, qué efecto de borde estilizado desea aplicar sobre las celdas, si deben aplicar éstas un efecto de degradado o no, qué tipo de letra (y con qué tamaño) desea que aparezca en las vistas de traza y código, o incluso la distancia a la que deben aparecer unas celdas respecto a otras.

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 y qué nivel de zoom hay que aplicar tanto sobre el árbol como sobre la pila.

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 flags 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 parametrización de la visualización, cumpliendo los objetivos propuestos, poder recuperar la visualización íntegramente y además obtener la configuración exacta que tenía el programa cuando ésta fue guardada.

A continuación se proporciona la DTD que define este documento XML que genera SRec cada vez que un usuario decide guardar una visualización en este formato.

```
<!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,Grosor,
Grosor,Distancia,Distancia,Tipo,Tipo,Color,Color,Color,Color,Color,modoColor,modoColorDegr1,
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|flech
acodigoPR|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)*>
```

6.2.2. 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 permanece intacta mientras el usuario no decida modificarla explícitamente, es la configuración que el programa carga al arrancar cada sesión.

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. La DTD que define la estructura del documento XML que alberga todas estas opciones de configuración del programa es la siguiente:

```
<!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,colorPanel,c
olorActual,colorCActual,
varios,colorPalabrasReservadas,colorComentarios,colorMetodoForeground,colorMetodoBackground,
colorCodigo,colorIluminado,fuentesTrazaCodigo,zoomsDefecto,distancias,colorm2_0,colorm2_1,co
lorm2_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)>
```

6.2.3. Documento XML para los parámetros de un algoritmo

Otro de los documentos XML que genera SRec está destinado a almacenar parámetros 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.

La DTD que define la estructura de estos documentos XML de gran utilidad es la que se muestra a continuación:

```
<!ELEMENT valoresAlgoritmoVisualizador (algoritmo)>
<!ELEMENT algoritmo (parámetro+)>
<!ATTLIST algoritmo nombre CDATA #REQUIRED>
<!ELEMENT parametro (EMPTY)>
<!ATTLIST parametro val CDATA #REQUIRED>
<!ATTLIST parametro clase CDATA #REQUIRED>
```

6.2.4. 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.

Este documento XML se genera con el paquete externo ToXML (programado por Harsh Jain [16]), motivo por el cual no se proporcionará su DTD, dada además la alta complejidad de la misma.

6.3. Integración de JGraph

La integración de JGraph en la arquitectura de SRec resultó tremendamente sencilla gracias a que los componentes gráficos que implementa son extensiones de los componentes que contiene la librería Swing, que ya venía siendo utilizada con anterioridad en el desarrollo del programa.

Es por ello que se han utilizado “clases puente” dentro del paquete Gráfica para la conexión de la interfaz (donde se alojan las vistas implementadas con JGraph) con

los datos que maneja el programa (en los que se basan para crear las vistas). Estas "clases puente" obtienen los datos de la visualización y manejan la librería JGraph para la correcta representación de los mismos siguiendo los pasos necesarios para desarrollar la vista adecuada.

Cada una de estas "clases puente" representa e implementa una de las vistas generada con JGraph. Así, SRec cuenta con las clases ContenedorArbol y ContenedorPila que se encargan, empleando los mismos datos y la misma librería, de mostrar diferentes visualizaciones.

Por tanto, JGraph ha permitido mantener la arquitectura y la filosofía establecida desde un principio, por lo que se ha mostrado como la librería ideal para este propósito.

En cuanto al nivel de explotación de JGraph que realiza SRec éste es bajo. La librería ofrece multitud de características y funcionalidades que apenas han sido explotadas por no ser necesarias para los objetivos del programa.

6.4. Sitio web de SRec

SRec cuenta con su propio sitio web para cumplir el objetivo marcado inicialmente de difundir el uso del programa entre diversas comunidades educativas. La web, para llegar al mayor número de personas posible, se presenta en español y en inglés, proporcionando toda la información necesaria acerca del programa (ver ilustración 43).

Así, ésta identifica a la Universidad Rey Juan Carlos y al Departamento de Sistemas y Lenguajes Informáticos I de la Escuela Técnica Superior de Ingeniería de Informática como responsables de este programa.

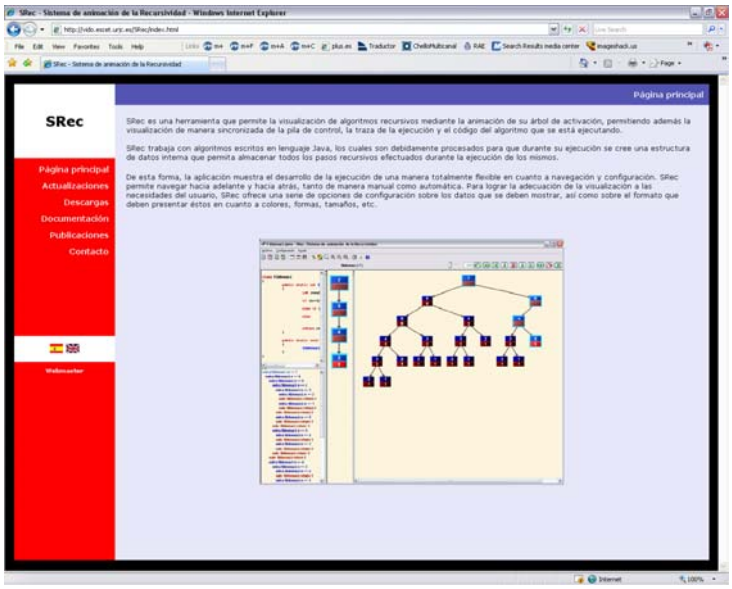

Ilustración 43 Web de SRec

Se proporciona a través de la web un conjunto de secciones que permiten conocer más a fondo el programa y acceder a cierta información relacionada con el mismo. Las secciones de las que consta la web son las siguientes:

- **"Actualizaciones"**: informa de las últimas novedades que han tenido lugar en la web y de los cambios que han supuesto, lo que permitirá saber si por ejemplo existe una nueva versión de SRec disponible o se ha añadido documentación nueva.
- **"Descargas"**: da acceso a la última versión disponible del programa, permitiendo su descarga por parte del visitante. También permite bajar ejemplos de animaciones, y otro tipo de material relacionado con el programa.
- **"Documentación"**: aporta amplia información sobre el programa. Se presenta el manual de usuario, que da una idea bastante amplia sobre cómo se debe manejar el programa para sacarle el mayor rendimiento posible. También se aporta cierta información sobre la concepción de SRec, su finalidad, su arquitectura...
- **"Publicaciones"**: aparecen disponibles para su descarga algunas publicaciones relacionadas con SRec dirigidas a Simposios y Conferencias.
- **"Contacto"**: se ofrecen medios de contacto electrónico con los responsables del proyecto.

7. Evaluaciones de usabilidad

SRec ha sido objeto de tres evaluaciones de usabilidad realizadas entre el alumnado de las asignaturas “Estructura de datos y algoritmos avanzados” (curso 2006/2007) y “Diseño y análisis de algoritmos” (cursos 2007/2008 y 2008/2009).

El objetivo de tales pruebas era conocer la facilidad de aprendizaje que presenta el uso del programa en cuanto al manejo, la gestión de la configuración del formato, las opciones de visualización disponibles y la carga y almacenamiento de visualizaciones, lo cual da al alumnado la posibilidad de llevar en formato electrónico ejemplos y ejercicios de un lugar a otro (de casa a clases prácticas, y viceversa, por ejemplo) para poder seguir utilizando las visualizaciones sin necesidad de generarlas y configurarlas de nuevo.

También se pretendía medir la capacidad real del programa a la hora de asistir en el aprendizaje de los alumnos sobre la materia impartida en clase, para comprobar si facilita las tareas de análisis y depuración de algoritmos.

Se consiguió información muy valiosa de estas dos pruebas, que sirvió fundamentalmente para continuar el desarrollo de la aplicación siguiendo un camino afín a las necesidades reales mostradas por el colectivo estudiantil.

Antes de la finalización de su desarrollo, SRec también empezó a ser empleado para la docencia dentro de la asignatura “Diseño y análisis de algoritmos” por parte de D. Ángel Velázquez Iturbide durante el curso 2007/2008 dentro del capítulo “Optimización de algoritmos”.

7.1. Organización

Las pruebas tuvieron una duración de dos horas cada una y consistieron en la realización de una serie de ejercicios relacionados directamente con la materia impartida (ver Anexo).

En los enunciados que se repartieron se proponían varios ejercicios. Los primeros se orientaban a lograr la familiarización del alumno con el programa y todas

sus opciones de configuración y funcionalidad. Los ejercicios posteriores ahondaban en cuestiones relacionadas con el análisis de algoritmos.

El primero de ellos siempre era realizado por el profesor con el ánimo de introducir el programa a los alumnos, indicando su modo de carga, y un caso básico de uso (generación de una visualización y presentación de opciones). Los siguientes ejercicios, también orientados a la introducción al manejo de la herramienta, eran realizados por los alumnos, asistidos por los profesores cuando presentaban dudas sobre los objetivos de los ejercicios o sobre el funcionamiento del programa.

Tras la realización de todos los ejercicios planteados, los alumnos respondían a un sencillo cuestionario sobre usabilidad, utilidad y calidad de SRec, en el que además podían aportar ideas para mejorar el programa, aún en fase de desarrollo.

7.2. Evaluación 1

La primera de las pruebas tuvo lugar el 24 de mayo de 2007, se contó con la participación de 7 alumnos de la asignatura “Estructura de datos y algoritmos avanzados”, impartida por Carlos A. Lázaro Carrascosa.

Se les pidió como ejercicio final de la práctica la depuración de una versión del algoritmo *mergesort* que contenía dos errores.

Los resultados de la prueba permitían obtener la conclusión de que el programa necesitaba pequeñas mejoras en la interfaz. Así, algunos alumnos no fueron conscientes de algunas de las opciones que proporcionaba el programa como por ejemplo la edición de código, utilizando otro software ajeno para esa tarea, y es que, al fin y al cabo, SRec no está orientado a la depuración de programas, sino a la visualización de algoritmos ya programados.

Tan sólo tres estudiantes de los siete que realizaron la prueba resolvieron correctamente el ejercicio. Como nota global, SRec obtuvo un 4,14 sobre 5. Respecto a usabilidad, SRec obtuvo un buen 4,43 sobre 5. A continuación se presenta una tabla con los valores medios obtenidos en los cuestionarios junto a algunos de los comentarios recibidos sobre SRec:

SRec es fácil de usar	3,88
SRec me ha ayudado a analizar algoritmos recursivos para buscar el error	2,63
SRec me ha ayudado a analizar algoritmos recursivos para comprobar que la solución propuesta es la correcta	4,13
Calidad general de SRec para analizar la recursividad	3,38
Calidad del menú principal	3,75
Calidad de los controles de animación	4,38
Calidad de la vista de la traza	3,75
Calidad de la vista de la pila de control	4,00
Calidad de la vista del árbol de activación	4,25
Calidad de la configuración de las visualizaciones	3,88
Interacción con los paneles	3,63
SRec me ha gustado	3,63

Tabla 1 Valores medios de la primera sesión de evaluación

Algunas de las críticas recibidas en esta primera sesión se centraron en la vista de traza, que aportaba insuficiente información para el objetivo que tenían que conseguir. Otras críticas se centraron en la vista estática de código, precisamente por su condición no dinámica. Algunos estudiantes reclamaron que se señalara qué línea de código se ejecutaba en cada momento, si bien es cierto que los pasos de las animaciones no representan la ejecución de ninguna sentencia de código sino la de una subllamada.

7.3. Evaluación 2

La segunda prueba fue desarrollada el 4 de diciembre de 2007 dentro de la asignatura “Diseño y análisis de algoritmos”, impartida por Ángel Velázquez Iturbide. Fue de carácter similar, y con una intencionalidad igualmente parecida a la de la primera prueba.

Se les propuso a los 28 alumnos asistentes a la sesión una serie de ejercicios que tenían que realizar. Como ejercicio final se le pidió a los alumnos que analizaran el algoritmo de recursividad múltiple del problema de competición para entregar posteriormente dos grafos: uno del árbol de recursión (copiado desde la pantalla de SRec) y otro del grafo de dependencia (ver Anexo).

Los resultados de la prueba, que se hicieron sobre una versión más avanzada del programa habiendo tenido en cuenta los resultados de la prueba anterior, mostraron un carácter muy positivo. Las mejores opiniones sobre el programa se vertieron sobre la vista del árbol de activación, gracias a su diseño estilizado y altamente configurable y a que ayuda a entender fácilmente el proceso recursivo, tal y como indicó un notable número de alumnos.

Respecto a los resultados académicos de las pruebas, 26 de los 28 alumnos entregaron correctamente el ejercicio, y de éstos, 23 emplearon SRec para la resolución del mismo. En esta ocasión, la nota sobre usabilidad subió hasta el 4,5 sobre 5, mientras que la nota final del programa alcanzó los 4,26 puntos sobre 5. A continuación se adjunta una tabla que repasa las puntuaciones obtenidas por el programa:

SRec es fácil de usar	4,50
SRec me ha ayudado a analizar algoritmos recursivos para analizar qué llamadas se realizan en tiempo de ejecución	4,29
SRec me ha ayudado a analizar algoritmos recursivos para identificar la dependencia entre llamadas	4,36
Calidad general de SRec para analizar la recursividad	4,29
Calidad del menú principal	4,07
Iconos	3,86
Calidad de los controles de animación	4,50
Calidad de la vista de la traza	4,00
Calidad de la vista de la pila de control	4,04
Calidad de la vista del árbol de activación	4,43
Calidad de la configuración de las visualizaciones	3,82
Interacción con los paneles	3,89
SRec me ha gustado	4,26

Tabla 2 Valores medios de la segunda sesión de evaluación

Las críticas recibidas se referían en muchos casos a meros aspectos técnicos, como la exportación a formato GIF animado, que fue posteriormente mejorada, la imposibilidad de exportar una única imagen simple en formato JPG, posibilidad que se introdujo posteriormente para los formatos GIF, JPG y PNG, o la inexistencia de un mapa de navegación, que fue insertado más tarde para facilitar la navegación por árboles de gran tamaño, sobre todo en cuanto a anchura se refiere.

7.4. Evaluación 3

La tercera prueba fue realizada por 22 estudiantes el 21 de noviembre de 2008, también dentro de la asignatura impartida por Ángel Velázquez Iturbide. Tras los ejercicios de introducción a la aplicación se les propuso un ejercicio en el que debían eliminar la redundancia existente en el algoritmo de competición introducida por la recursividad múltiple. Además de eso, se les pidió la entrega de un árbol de recursión significativo del algoritmo, copiándolo desde la ventana de la aplicación y el grafo de dependencia asociado.

Respecto a los resultados académicos de la práctica, se dobló el número de sobresalientes conseguidos respecto a prácticas anteriores (de similar dificultad) y muchos de los alumnos incluyeron en el informe de prácticas de manera espontánea comentarios sobre que SRec les había ayudado a abordar el ejercicio y a entender mejor el funcionamiento de ciertos algoritmos.

SRec es fácil de usar	4,20
SRec me ha ayudado a determinar las llamadas recursivas que se realizan en tiempo de ejecución	4,19
SRec me ha ayudado a identificar las dependencias entre llamadas recursivas	4,05
Calidad general de SRec para analizar la recursividad	4,00
Iconos	3,57
Calidad de los controles de animación	3,71
Calidad de la vista del árbol de activación	4,00
Calidad del visor de árboles grandes	3,86
Calidad de la configuración de formatos	3,76
Calidad de la configuración de zoom	3,71
Calidad de la interacción con los paneles (croll, mover/mostrar/ocultar paneles...)	3,80
Proceso de generación de una animación	4,00
Proceso de carga/almacenamiento de una animación	4,14
Visualización almacenada en un fichero de captura	4,10
SRec me ha gustado	3,95

Tabla 3 Valores medios de la tercera sesión de evaluación

La aceptación de la aplicación fue bastante alta por parte de los alumnos, muy receptivos a la adopción de herramientas que les faciliten el proceso de aprendizaje.

8. Conclusiones y trabajos futuros

La informática, disciplina orientada a la gestión automatizada de la información, está destinada a ofrecer servicios que faciliten las tareas humanas y que abran además nuevas posibilidades de negocio e investigación. En el contexto de la investigación orientada a la informática educativa, SRec nace para satisfacer una demanda, la de la generación automática de visualizaciones animadas, facilitando la creación de ejemplos para el personal docente y el proceso de aprendizaje para el alumnado.

Esta funcionalidad se alcanza, precisamente, con la gestión automática de la información que generan los algoritmos recursivos que ejecuta el programa. Tal información es convenientemente creada, manipulada y almacenada en formatos basados en XML y recuperada para la generación de visualizaciones con posibilidad de animación.

SRec es, por tanto, un buen ejemplo del uso de la informática aplicada a un problema concreto. Un problema que, a la vista de los resultados de las evaluaciones de usabilidad realizadas entre el alumnado, la aceptación conseguida en varios congresos en los que ha sido presentado, y el interés suscitado entre el personal docente de alguna universidad española como la de Extremadura, ha sido satisfecho de manera notable.

El tiempo de desarrollo total de SRec comenzó el 11 de septiembre de 2006 y finalizó el pasado 20 de marzo de 2009, lo que supone un periodo de dos años y seis meses, interrumpido durante agosto de 2007, agosto de 2008 y un periodo estimado superior a seis meses que se dedicó al desarrollo de otros trabajos.

En ese tiempo se han desarrollado 85 clases junto a otras 69 externas, repartidas todas ellas en 13 paquetes (3 de ellos externos). Las clases propias suman un total de casi 38.000 líneas de código. Se han diseñado 24 iconos, cinco DTD para documentos XML, tres evaluaciones de usabilidad entre alumnos y 23 archivos HTML para la ayuda en cada uno de los idiomas en que se ofrece.

Además, durante este tiempo se han escrito siete publicaciones que se listan a continuación:

- L. Fernández, A. Pérez, Á. Velázquez, J. Urquiza. "*A framework for the automatic generation of algorithm animations based on design techniques*". Creating New Learning Experiences on a Global Scale – EC-TEL 2007, E. Duval, R. Klamma y M. Wolpers (eds.), Springer-Verlag (ISBN 3-540-75194-7), Lecture Notes in Computer Science (ISSN 0302-9743), 4753: 475-480, 2007.
- Á. Velázquez, A. Pérez, J. Urquiza. "*SRec: An animation system of recursion for algorithm courses*". ACM SIGCSE Bulletin, 40(3), aceptado, septiembre 2008 (ISSN 0097-8418). También en Proceedings of the 13rd Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008), ACM Press, 2008, aceptado.
- L. Fernández, A. Pérez, Á. Velázquez, J. Urquiza. "*SRec: Un sistema para la animación de árboles de recursión*". Actas del Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación (SINTICE 2007), Isabel Fernández de Castro (ed.), Thomson, 2007 (ISBN 978-84-9732-597-4), 215-222.
- Á. Velázquez, A. Pérez, C. Lázaro, J. Urquiza. "*Un sistema con múltiples vistas para la visualización y animación de la recursividad*". Actas do IX Simpósio Internacional de Informática Educativa (SIIE 2008), Escola Superior de Educação do Instituto Politécnico do Porto, 2007 (ISBN 978-972-8969-04-2), 331-332.
- A. Pérez. SRec: Manual de uso. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), nº 2008-03, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, mayo de 2008.
- Á. Velázquez, A. Pérez. "*SRec, software de animación de la recursividad*". Actas de las XV Jornadas de Enseñanza Universitaria de la Informática. Aceptado.
- Á. Velázquez, A. Pérez. "*Animación automatizada de técnicas de diseño de algoritmos*". Actas de las XV Jornadas de Enseñanza Universitaria de la Informática. Aceptado.

Durante el desarrollo de SRec se ha ahondado en la motivación de su creación y se ha visto la necesidad de desarrollar otras herramientas que sigan la estela de SRec, orientadas a la visualización de algoritmos basados en otras técnicas de diseño. En esta línea, durante el citado periodo de trabajo se comenzó el desarrollo de una segunda aplicación que genera dos vistas principales para algoritmos basados en la técnica de Divide y Vencerás, lo que ha tenido como primer fruto la quinta de las publicaciones que se listaron anteriormente.

Este segundo programa, cuyo desarrollo continuará próximamente de manera integrada en SRec, se enmarca dentro de la intención de desarrollar múltiples vistas específicas que permitan visualizar varias técnicas de diseño más, como la de Programación Dinámica. Su interfaz se puede apreciar en la ilustración 44.

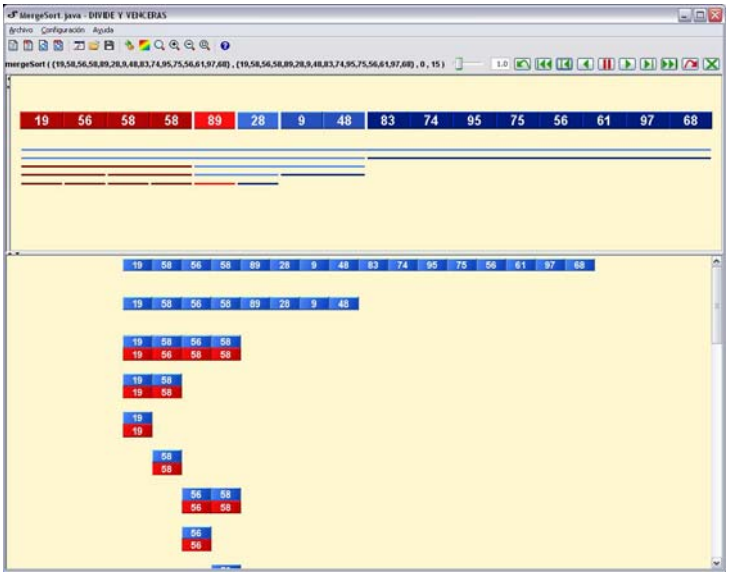

Ilustración 44 Segundo Programa orientado a la visualización

El desarrollo de vistas específicas para otras técnicas de diseño de algoritmos ha dado lugar a una publicación más:

- J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, y Jaime Urquiza-Fuentes. "*A design of automatic visualizations for divide-and-conquer algorithms*". En *Proc. V Program Visualization Workshop*, aceptado.

No obstante, no acaban aquí los planes para SRec, pues está en estudio dotar al programa de la capacidad de exportar visualizaciones a determinados lenguajes estándar de visualización de algoritmos, dejando la puerta abierta además a que SRec pueda también importar estos tipos de archivo para poder generar animaciones a partir de ellos.

Con esta funcionalidad, SRec se puede convertir en un interesante generador de visualizaciones en formato estándar que otros programas podrán utilizar cómodamente, facilitando así la transmisión de conocimiento entre diversas comunidades científicas y educativas.

Bibliografía

A continuación se lista una serie de referencias bibliográficas correspondiente a material consultado durante el desarrollo de este programa, bien para profundizar en la visualización de algoritmos, bien para conocer más a fondo algunas tecnologías empleadas en la implementación de la aplicación.

Libros

[1] Stasko, J.T., Domingue, J., Brown, M.H., y Price, B.A. (eds.). "*Software Visualization"*. MIT Press, Cambridge, Massachusetts, 1998.
ISBN 0-26219-395-7

[2] T. L. Naps, G. Roessling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger y J. Á. Velázquez Iturbide. "*Exploring the role of visualization and engagement in computer science education*". *ACM SIGCSE Bulletin* 35, 2 (2003), pp. 131-152.
ISSN 0097-8418

[3] Fernández-Muñoz, L., y Velázquez-Iturbide, J.Á. "*Estudio sobre la visualización de las técnicas de diseño de algoritmos*". En *Proc. VII International Conf. Human-Computer Interaction (Interacción 2006)*, Universidad de Castilla-La Mancha, pp. 315-324.

[4] Eckel, Bruce. "*Piensa en Java*", Pearson Prentice Hall (2007).
ISBN 978-8-48966-034-2

[5] McLaughlin, Brett. "*Java and XML*", O'Reilly (2001).
ISBN 0-59600-197-5

[6] Goldfarb, Charles F. "*Charles F. Goldarb's XML handbook*", Prentice Hall PTR (2004).
ISBN 0-13049-765-7

[7] Brassard, Gilles. "*Fundamentos de algoritmia*", Pearson Prentice Hall (2006).
ISBN 8-4966-000-X

[8] Meyer, Bertrand. "*Construcción de software orientado a objetos*", Prentice Hall (2002).
ISBN 8-48322-040-7

[9] Peña Marí, Ricardo. "*Diseño de programas, formalismo y abstracción*". Pearson Prentice Hall (2004).
ISBN 8-42054-191-5

[10] Eriksson, Hans-Erik. "*UML 2 toolkit*". Wiley Publishing
ISBN 0-47146-361-2

[11] Gamma, Erich. "*Design Patterns: Elements of Reusable Object-Oriented Software*". Addison Wesley (2000).
ISBN 0-20163-341-2

[12] Miano, John. "*Compressed image file formats JPEG, PNG, GIF, XBM, BMP*". ACM Press (1999).
ISBN 0-20160-443-4

Páginas web

[13] Darugar, Parand Tony. "*Effective XML processing with DOM and Xpath in Java*". IBM
http://www.ibm.com/developerworks/xml/library/x-domjava/
(último acceso 30-mayo-2008)

[14] Le Hégaret, Philipe. "*Document Object Model (DOM)*". W3C
http://www.w3.org/DOM/
(último acceso 30-mayo-2008)

[15] Equipo JGraph. "*JG*raph User Manual". JGraph
http://jgraph.com/pub/jgraphmanual.pdf
(último acceso 30-mayo-2008)

[16] Harsh Jain. "*Java2XML: A Java To XML Converter*". Java.net
https://java2xml.dev.java.net
(último acceso 30-mayo-2008)

Anexo: Enunciados de las sesiones de evaluación

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 pàra 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:
    _____
    _____

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:

6. **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.

7. **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:

i. Procesar la clase *ClaseEvaluacion* disponible en el sitio *web* de la asignatura (menú de Archivo).
ii. Generar una animación (menú de Archivo).
iii. Experimentar con todos los controles de animación hasta que se comprenda su funcionamiento.
iv. Almacenar la animación y cargarla de nuevo (ambas operaciones, en menú de Archivo).

8. **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:

i. Generar una animación.
ii. 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.
iii. 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).

9. **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.

10. **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:

13. Di qué características te parece que podrían ser **útiles** pero SRec carece de ellas:
    _____
    _____
    _____

14. Di qué características de SRec te parecen tan **poco útiles** que las suprimirías:
    _____
    _____
    _____

15. Describe los **aspectos positivos** que encuentras en SRec (sobre todo si no se han mencionado antes):
    _____
    _____
    _____

16. 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 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:

11. **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.

12. **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:

i. Procesar la clase *ClaseEvaluacion* disponible en el sitio *web* de la asignatura (menú de Archivo).
ii. Generar una animación (menú de Archivo).
iii. Experimentar con todos los controles de animación hasta que se comprenda su funcionamiento.
iv. Almacenar la animación y cargarla de nuevo (ambas operaciones, en menú de Archivo).

13. **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:

i. Generar una animación.
ii. 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.
iii. 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).

14. **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).

15. **Cuestionario.** Duración aproximada: 20 minutos.

El alumno debe responder a las preguntas del cuestionario que se entregará en papel.

Cuestionario

Nombre y apellidos (opcional): _____

En las perguntas 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/coultar 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 encuentreas 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)
   _____
   _____
   _____