Sobre la interacción en la visualización del software

J. Ángel Velázquez Iturbide, Antonio Pérez Carrasco, Jaime Urquiza Fuentes, Francisco J. Almeida Martínez
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos
28933 Móstoles, Madrid
{angel.velazquez,antonio.perez.carrasco.jaime.urquiza,francisco.almeida}@urjc.es

Resumen

Un aspecto que ha recibido relativamente poca atención en la visualización del software es la interacción con el usuario. Generalmente se ha limitado a su animación o a facilidades educativas, formas de interacción adecuadas para la comprensión de programas. Sin embargo, se necesitan formas de interacción más avanzadas para tareas de usuario más complejas, como el análisis de programas. En la comunicación proponemos una modificación de una taxonomía de la interacción en el campo de la visualización de la información, adaptada a la visualización del software. La nueva taxonomía recoge una definición más precisa de las categorías de interacción en términos de elementos de programación, así como la dimensión del tiempo. La taxonomía se utiliza para caracterizar varios sistemas de visualización.

1. Introducción

Una de las tecnologías que más interés ha despertado en las últimas décadas para la enseñanza de la programación es la visualización del software. Inicialmente el esfuerzo de investigación se centró en aspectos técnicos de la generación de visualizaciones, pero enseguida se constató que el uso de visualizaciones no siempre producía una mejora educativa.

El análisis de evaluaciones realizadas ha permitido deducir que es más importante el uso docente que se haga de las visualizaciones que ellas mismas [5]. En esta línea, se ha propuesto una taxonomía de niveles de implicación de los alumnos [7], sobre la hipótesis de que una mayor implicación del alumno produce un mayor éxito educativo. Esta taxonomía se ha utilizado en la docencia, existiendo constancia de numerosas evaluaciones positivas [11].

En esta comunicación pretendemos profundizar en el papel de la interacción persona-ordenador en la visualización del software. La sección segunda presenta antecedentes de este trabajo, incluyendo definiciones y el papel de la interacción en la visualización. En la tercera sección realizamos una propuesta de taxonomía de las interacciones con visualizaciones del software.

2. Antecedentes

En primer lugar, presentamos algunas definiciones relacionadas con la visualización, para centrar nuestro análisis. Después presentamos el tratamiento dado a la interacción con el usuario en visualización de la información y del software.

2.1. Definiciones

La *visualización de información* es un campo de la interacción persona-ordenador. Por concreción y brevedad, nos referimos a la definición que dan Card, Mackinlay y Shneiderman [2]:

> "El uso de representaciones generadas por ordenador, visuales e interactivas, de datos abstractos para amplificar la cognición".

Obsérvese que no solamente se destaca el uso del ordenador y su naturaleza visual (que son evidentes) sino también su carácter interactivo.

El término *visualización del software* [3][10] alude a la visualización de los diversos aspectos del software (código, diseño, pruebas, etc.). Si nos centramos en la programación a pequeña escala, los términos más extendidos son *visualización de programas* y *visualización de algoritmos*. El primer término engloba la visualización de elementos ligados al código fuente, sea código o datos, mientras que el segundo recoge una visualización de más alto nivel, no necesariamente

ligado a un código concreto. Con frecuencia se utiliza el término *visualización del software* (VS) en este sentido más específico para la programación a pequeña escala. A falta de otro término mejor, así lo usamos en la comunicación.

También ocupa un papel destacado la distinción entre visualización estática o dinámica del software, llamándose *animación* la segunda. En la comunicación utilizaremos generalmente el término *visualización* para referirnos a ambas.

2.2. Interacción en visualización del software

Existen diversas taxonomías para el estudio y caracterización de los sistemas de visualización de software. Algunas taxonomías hacen distinciones similares a las de la subsección anterior. Otras taxonomías consideran el método de creación o de especificación de las animaciones. Nos centramos aquí en las taxonomías que dan un papel más destacado a la interacción, las taxonomías de Roman y Cox [9] y de Price, Baecker y Small [8].

Ambas taxonomías proponen categorías parecidas, basadas en un modelo ideal de producción y de roles en VS (véase Tabla 1).

Tabla 1. Categorías de las taxonomías de Roman y Cox, y de Price, Baecker y Small

Roman y Cox	**Price, Baecker y Small**
–	Ámbito
Ámbito	Contenido
Abstracción	Método
Método de especificación	
Interfaz: vocabul. gráfico	Forma
Presentación	
Interfaz – interacción	Interacción
–	Eficacia

En lo que se refiere a la categoría de interacción, ambas taxonomías coinciden. Roman y Cox definen indirectamente la categoría de interfaz mediante la pregunta "¿qué facilidades proporciona el sistema para la presentación visual de la información?". La subcategoría de interacción incluye las herramientas que el usuario puede usar para modificar la presentación. Consideran que hay dos formas de interacción:

- Mediante controles.
- Mediante la imagen.

En definitiva, Roman y Cox consideran técnicas de interacción a un bajo nivel, que pueden usarse para diversos fines.

La taxonomía de Price, Baecker y Small [8] define indirectamente la interacción mediante la pregunta "¿cómo puede el usuario del sistema de VS interactuar con él y controlarlo?". A su vez, distinguen 3 subcategorías de interacción:

- Estilo.
- Navegación:
  - Control de ocultación ("elisión").
  - Control temporal:
    - Dirección.
    - Velocidad.
- Facilidades de grabación y reproducción.

En su taxonomía, Price, Baecker y Small incluyen tres aspectos diferentes. El estilo se refiere a técnicas de interacción de bajo nivel, y es similar a la interacción de Roman y Cox. La navegación incluye algunas formas de interacción para navegar en espacio o tiempo. Finalmente, las facilidades de grabación y reproducción engloban funciones de exportación e importación, generalmente con fines docentes.

Otra taxonomía distinta es la de lenguajes de animación de algoritmos, debida a Karavirta *et al.* [6]. Su objetivo declarado es servir de complemento de la taxonomía de Price, Baecker y Small, ya que ésta no caracteriza lenguajes sino sistemas. Sin embargo, difícilmente puede cumplir este papel ya que su ámbito es menos general, al limitarse a visualización de algoritmos.

Proponen cuatro categorías: visualización, dinámica, interacción del usuario y metalenguaje. Las dos primeras coinciden básicamente con la categoría "forma" de Price, Backer y Small. A su vez, la categoría tercera tiene cuatro subcategorías: control, responder, cambiar y anotar.

Los autores reconocen que su categoría de interacción del usuario está influida por la taxonomía de niveles de implicación [7]. En ésta se distinguen 6 formas (no jerarquizadas) de implicación de los alumnos con las animaciones:

- Sin ver.
- Ver.
- Responder.
- Cambiar.

- Construir.
- Presentar.

Estas dos últimas taxonomías se centran en tareas de usuario, sin poner énfasis en la propia interacción que deben realizar. En consecuencia, resultan de poca utilidad para nuestro objetivo.

2.3. Interacción en visualización de la información

En visualización de la información también encontramos numerosas taxonomías. Por concreción y falta de espacio, nos limitaremos a la taxonomía de Yi *et al.* [17]. En este mismo trabajo puede encontrarse una recopilación de taxonomías, que se centran en técnicas de interacción o en tareas de usuario.

Yi *et al.* consideran que conviene unir ambos enfoques. Por tanto, proponen el criterio de "intención del usuario", es decir: "¿qué quiere conseguir el usuario?". Este criterio permite agrupar técnicas de interacción de bajo nivel y relacionar las tareas del usuario con la interacción.

Se distinguen siete categorías de interacción. Siguiendo el estilo de sus autores, para cada categoría describimos la intención del usuario mediante una frase coloquial y una definición.

- Seleccionar: marcar algo como interesante. Las técnicas de selección proporcionan a los usuarios la capacidad de marcar elementos de interés para mantener su rastro.
- Explorar: mostrar algo más. Las técnicas de exploración permiten a los usuarios examinar un subconjunto distinto de datos.
- Reconfigurar: mostrar una disposición distinta. Las técnicas de reconfiguración proporcionan a los usuarios distintas perspectivas del conjunto de datos mediante cambios en la disposición espacial de las representaciones.
- Codificar: mostrar una representación distinta. Las técnicas de codificación permiten a los usuarios alterar la representación visual fundamental de los datos, incluyendo la apariencia visual de cada elemento (p.ej. color, tamaño y forma).
- Abstraer/desarrollar: mostrar más o menos detalle. Las técnicas de abstracción/desarrollo proporcionan a los usuarios la capacidad de ajustar el nivel de abstracción de una representación.
- Filtrar: mostrar algo condicionalmente. Las técnicas de filtrado permiten a los usuarios cambiar el conjunto de elementos que se presentan según condiciones concretas.
- Conectar: mostrar elementos relacionados. Conectar se refiere a técnicas de interacción que se usan para (1) resaltar asociaciones y relaciones entre elementos que ya están representados, y (2) mostrar elementos ocultos que son importantes para algún elemento.

Puede observarse que la taxonomía es muy general con categorías descriptivas y de alto nivel.

3. Propuesta de interacción en visualización de software

Comenzamos esta sección señalando las dificultades encontradas al aplicar la taxonomía de Yi *et al*. Después, proponemos una modificación de su taxonomía, adaptada al software.

3.1. Experiencia y dificultades con taxonomías

Los autores de la comunicación han desarrollado sistemas de visualización muy diversos. VAST es un sistema de animación del análisis sintáctico de programas [1]. WinHIPE es un entorno de programación que permite animar la evaluación de expresiones de un lenguaje funcional [13]. GreedEx es un ayudante interactivo para experimentar con la optimidad de varios algoritmos voraces [14]. SRec es un sistema de animación de la recursividad en Java [16].

Los autores han utilizado en diversas ocasiones las taxonomías de visualización del software para caracterizar los sistemas. Algunos aspectos han quedado claramente caracterizados, mientras que otros no, especialmente la interacción. También han caracterizado la interacción en estos sistemas con taxonomías de visualización de la información, con mejores resultados [15], pero no totalmente satisfactorios.

Al aplicar la taxonomía de Yi *et al.*, se han encontrado los siguientes problemas:

- Ausencia de tratamiento de la dimensión tiempo. En efecto, las visualizaciones de la

información suelen centrarse en datos complejos, difíciles de examinar, pero estáticos en el tiempo.
- Ambigüedad al identificar la categoría a la que pertenece una clase de interacción. Hemos encontrado ambigüedad en varios casos:
  - "Codificar" engloba cambios de representación gráfica muy variados, desde un mero cambio de configuración a un cambio de vista. Entre estos dos casos extremos, otras categorías también tienen un efecto sobre la representación (reconfigurar, abstraer/desarrollar, filtrar). Por tanto, se necesita una definición más precisa de "codificar".
  - La intención de la categoría "abstraer/ desarrollar" es ver algo con más detalle pero, según su definición, debe variar el nivel de abstracción. Ambas frases son contradictorias, como queda patente al incluir el zoom geométrico: no produce cambios en el nivel de abstracción ni permite ver más elementos, sino que permite verlos mejor. Ver más detalles correspondería a eliminar un filtro o a desarrollar el nivel de abstracción. Por tanto, el zoom geométrico sería una técnica de "explorar".

3.2. Propuesta de taxonomía modificada para visualización del software

Proponemos una modificación de la taxonomía con dos dimensiones:

- Intención del usuario.
- Espacio/tiempo.

Con respecto a la dimensión de la intención del usuario, cada categoría debe definirse de forma precisa. Consideramos que una definición es precisa si se realiza en términos de su efecto sobre elementos de programación.

Con respecto a la dimensión de espacio/ tiempo, la consideramos necesaria para recoger la especificidad de la visualización de software:

- El espacio. Actúa sobre una visualización estática, que representa el estado del programa, sus datos o ambos en un instante.
- El tiempo. Actúa sobre los distintos estados del programa, que pueden visualizarse mediante una animación o estar representados en la propia visualización estática.

La taxonomía queda tal y como recoge la Tabla 2. Obsérvese que hay dos categorías nuevas, "particularizar" y "representar", que resultan de partir la categoría "codificar" de Yi *et al*.

Tabla 2. Dimensiones y categorías de la taxonomía de la interacción en visualización del software

	Espacio	Tiempo
Seleccionar		
Explorar		
Reconfigurar		
Particularizar		
Representar		
Abstraer/desarrollar		
Filtrar		
Conectar		

Veamos una definición detallada de cada categoría de la taxonomía. Cada categoría se ilustra con una visualización generada por alguno de los cuatro sistemas citados al comienzo de la sección 3.1. Se han seleccionado visualizaciones representativas, sin que ello signifique que un sistema permita más o menos interacciones.

Seleccionar – espacio. "Marcar algún elemento como interesante". Esta categoría permite seleccionar algún elemento del estado del programa. Es igual que la propuesta de Yi *et al.* De forma general, la selección es una acción que suele preceder a otra, es decir, suele seleccionarse con algún fin.

Por ejemplo, SRec permite resaltar las llamadas recursivas con algún parámetro o cuyo resultado es igual a algún valor concreto. La Figura 1 muestra el árbol de recursión de *fib*(8), donde se han resaltado las llamadas a *fib*(3).

En el resto de la comunicación no buscamos que todas las visualizaciones se entiendan completamente, pero sí que se entienda la intención del usuario. De todas formas, presentamos ampliada la Figura 1 para que se vea mejor la estructura de los árboles de recursión, que también aparece en otras figuras. Cada nodo tiene dos campos, el superior con los parámetros y el inferior con el resultado de la llamada.

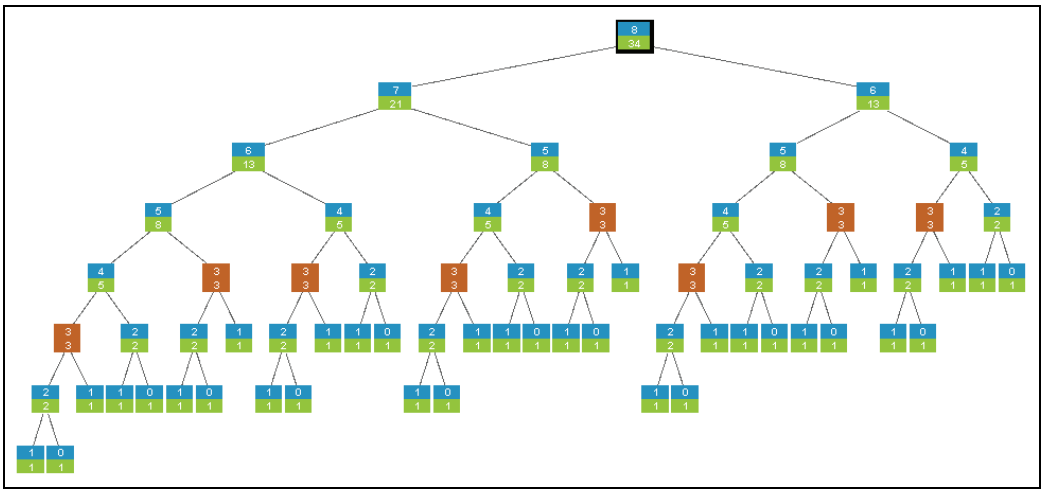

Figura 1. Árbol de recursión de *fib*(8), con las llamadas a *fib*(3) resaltadas

Seleccionar – tiempo. "Marcar algún instante como interesante". Esta categoría permite seleccionar un estado de la ejecución del programa. Se trata de una acción corriente en los depuradores, donde se marcan lugares del código como puntos de ruptura donde se parará la ejecución.

En la mayor parte de los sistemas de animación están predefinidos algunos instantes destacados: inicio, un paso adelante o atrás, final. En SRec, también puede seleccionarse una llamada recursiva terminada como paso previo a mover la animación a dicho instante.

Explorar – espacio. "Mostrar otros elementos". Esta categoría permite mostrar algunos elementos del estado del programa. Ejemplos de interacciones de esta categoría son *panning*+*zooming* o global+detalle.

La Figura 2 muestra la interacción global+detalle proporcionada por VAST. Puede observarse que la visualización externa proporciona la vista de detalle, mientras que la ventana inferior derecha contiene una vista global.

Explorar – tiempo. "Mostrar otros instantes". Esta categoría permite mostrar otros estados del programa, por lo que incluye las funciones de animación. Hay diversas formas estándar de navegar: adelante / atrás; un paso / varios pasos / saltar / hasta el final; automático / manual. Formas más avanzadas de navegar permiten seleccionar un instante que cumple ciertas características, como los puntos de ruptura en los depuradores.

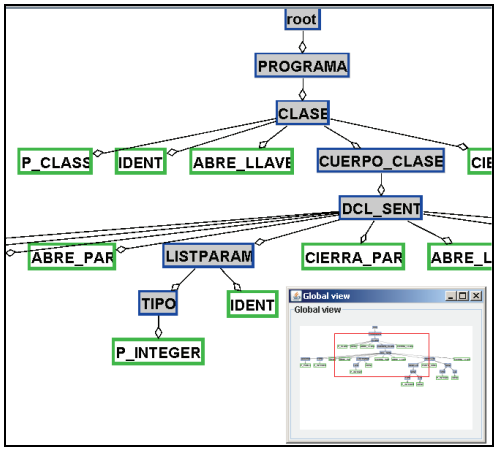

Figura 2. Visualización de un árbol de análisis sintáctico mediante una interfaz global+detalle

VAST incluye una barra de navegación muy completa que incluye los controles de animación estándar más la posibilidad de trasladarse a un punto concreto en la cadena de entrada procesada.

Reconfigurar – espacio. "Mostrar una disposición distinta de elementos". Esta categoría permite cambiar la disposición relativa de los

elementos. Esta disposición puede expresarse de forma espacial o mediante otros convenios.

Esta categoría tiene menor importancia en visualización del software que de la información, dado que el orden relativo de los instantes viene determinado por la semántica del lenguaje de programación. Sin embargo, hay sistemas, como WinHIPE, que permiten al usuario elegir distintas semánticas; otros sistemas incluso permiten ejecutar cualquier instrucción seleccionada.

GreedEx permite reconfigurar el orden relativo de los candidatos a tomar para un algoritmo voraz. Dicho orden relativo se expresa mediante tonos, de forma que los candidatos primeros se representan con tonos más oscuros. En la Figura 3, la primera visualización muestra las actividades en orden creciente de comienzo y en la segunda, en orden creciente de duración.

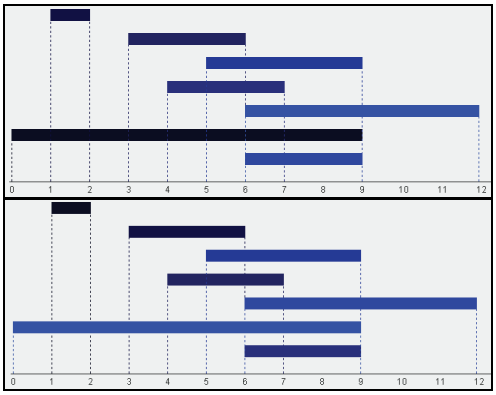

Figura 3. Reordenamiento mediante tonos de actividades a seleccionar

Reconfigurar – tiempo. "Mostrar una disposición distinta de los instantes". Esta categoría permite cambiar la disposición espacial de estados correspondientes a la ejecución del programa.

SRec permite reconfigurar el orden relativo de los valores iniciales y finales de vectores o matrices sobre los que operan algoritmos de divide y vencerás. La Figura 4 muestra dos reordenamientos del estado de subvectores en las llamadas recursivas de ordenación por mezcla. En la parte izquierda se sigue el orden estricto de comienzo y fin de las llamadas, mientras que en la parte derecha se unen espacialmente los estados de un subvector al iniciarse y acabar la llamada.

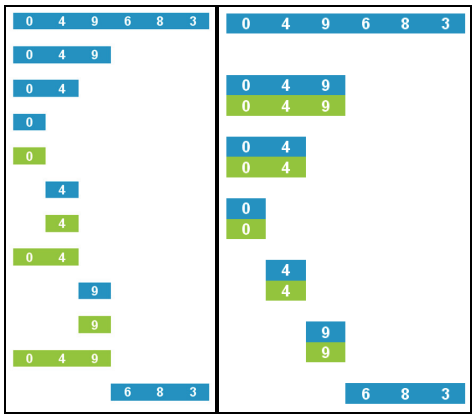

Figura 4. Reordenamiento de los distintos estados de entrada o salida de un vector

Particularizar – espacio. "Cambiar las propiedades visuales de sus elementos". Esta categoría respeta el contenido de la representación, cambiando los valores de sus propiedades visuales: color, tono, forma, orientación, etc.

SRec incluye un alto número de opciones de particularización: colores y tonos de distintos elementos, distancias horizontales y verticales, grosor y estilo de flechas y marcos, y fuentes de letra. También puede decidirse si el convenio de colores se usa para distinguir distintas clases de nodos o distintos métodos. La Figura 5 muestra el árbol de recursión de *fib*(5) con dos configuraciones distintas.

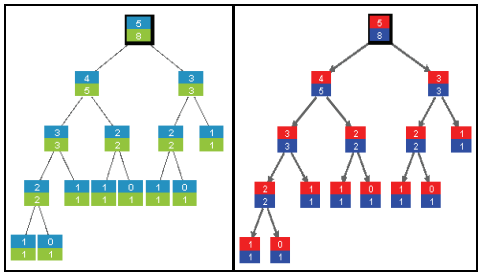

Figura 5. Árbol de recursión de *fib*(5) con dos configuraciones distintas

Particularizar – tiempo. "Cambiar las propiedades visuales de sus instantes". Esta categoría puede concretarse de varias formas, según se represente el tiempo. La forma más

evidente es mediante una animación, en cuyo caso pueden configurarse el modo (automático o manual, animación discreta, continua) y los efectos de animación. Otra forma de representar el tiempo es incluyendo elementos pasados en la propia visualización.

SRec permite particularizar el formato gráfico de los estados pasados. En la Figura 6, las llamadas recursivas terminadas se muestran tanto atenuadas como suprimidas.

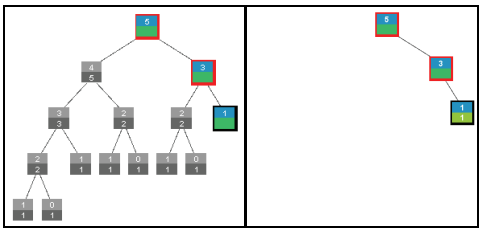

Figura 6. Representación de llamadas recursivas terminadas de forma atenuada y oculta

Representar – espacio. "Mostrar una representación distinta de los elementos". Esta categoría conlleva un cambio sustancial de la representación gráfica. Este "cambio sustancial" se concreta en representaciones con propiedades geométricas distintas. Si se ofrecen varias representaciones simultáneas, se habla de vistas múltiples.

SRec proporciona tres representaciones de la recursividad: árbol de recursión, pila de ejecución y rastro. La Figura 7 muestra parte del rastro generado para *fib*(8); compárese con la Figura 1.

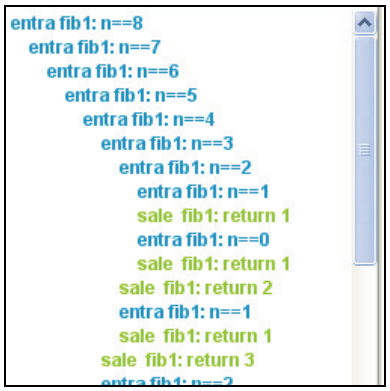

Figura 7. Rastro de las llamadas recursivas de *fib*(8)

Representar – tiempo. "Mostrar una representación distinta de los instantes". Quizá se trata de la categoría más novedosa. El tiempo puede representarse de varias formas, como hemos comentado en la categoría de "particularizar – tiempo".

Aunque no lo soporta explícitamente, SRec permite al usuario que elija entre representar el tiempo en la propia codificación del espacio (dando lugar a un árbol de recursión) o dejar simplemente la animación (dando lugar a la pila de ejecución). La elección tiene efectos sobre otras acciones: la primera opción permite al usuario seleccionar directamente cualquier llamada anterior, mientras que la segunda sólo permite navegar hasta alcanzar dicho estado.

Abstraer/desarrollar – espacio. "Mostrar los elementos en un diferente nivel de abstracción". Esta categoría permite manejar la complejidad del software en términos del nivel de abstracción de los elementos de programación. Si se baja el nivel de abstracción, aparecen más elementos; si se aumenta, el número de elementos disminuye.

SRec da una forma limitada de abstracción. Permite ver la relación estructural entre llamadas, como sucede en un árbol de recursión, pero también proporciona un resumen del tamaño del mismo (véase la Figura 8).

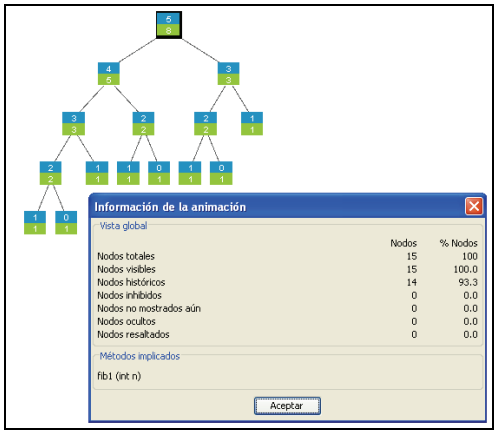

Figura 8. Árbol de recursión de *fib*(5) y resumen

Abstraer/desarrollar – tiempo. "Mostrar los instantes en un diferente nivel de abstracción". Esta categoría permite controlar la complejidad de

la ejecución del software en términos del desarrollo de los elementos de programación.

SRec contempla que una llamada recursiva se expanda o solamente muestre su resultado. La primera posibilidad, correspondiente a las figuras anteriores, muestra un rastro completo de una ejecución. La segunda posibilidad produce una visualización similar a la definición inductiva. En la Figura 9 se aprecia el efecto sobre *fib*(5).

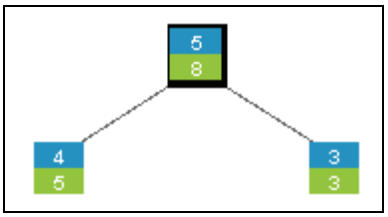

Figura 9. Árbol de recursión al nivel de abstracción de definición inductiva

Filtrar – espacio. "Seleccionar los elementos a mostrar mediante su identificación o condiciones". Esta categoría permite manejar la complejidad del software en términos de los elementos de programación mostrados a un cierto nivel de abstracción.

WinHIPE permite simplificar la visualización de una expresión funcional mediante una variante de la técnica de ojo de pez [12]. El resultado es un compromiso entre mostrar una visión global de la expresión y la parte más relevante (la subexpresión a evaluar). En la Figura 10 se muestra su efecto sobre una expresión intermedia del cálculo de un árbol binario a partir de sus recorridos en preorden y orden central.

El filtrado puede ir acompañado de operaciones complementarias para mostrar más información, como el zoom semántico. SRec lo proporciona para consultar todos los valores asociados a un nodo del árbol de recursión.

Filtrar – tiempo. "Seleccionar los instantes a mostrar mediante su identificación o condiciones". Esta categoría permite controlar la complejidad de la ejecución del software en términos de los elementos de programación mostrados a un cierto nivel de abstracción.

SRec permite mostrar, para cada llamada recursiva, sus valores de entrada, su salida o ambos. También permite seleccionar los métodos y parámetros a visualizar. En algoritmos de divide y vencerás, también puede optar entre una subestructura de datos o la estructura completa.

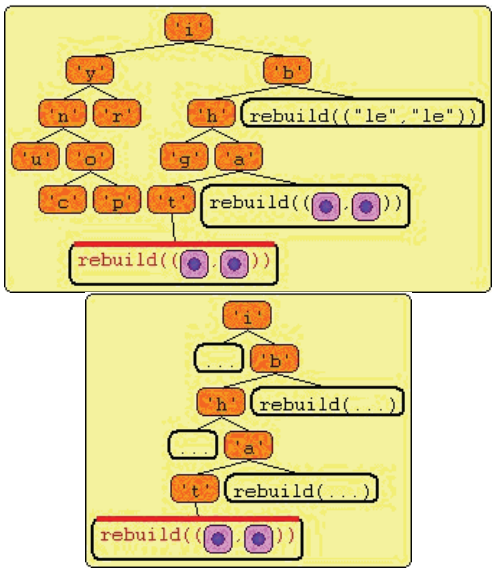

Figura 10. Una expresión completa y su simplificación mediante ojo de pez

En la Figura 11 se ve la diferencia de tamaño del árbol de recursión entre mostrar tres métodos distintos o solamente uno, en ambos casos coloreado en azul celeste. Además, solamente se muestran los parámetros de cada llamada, no su resultado. El algoritmo visualizado es el de ordenación por mezcla.

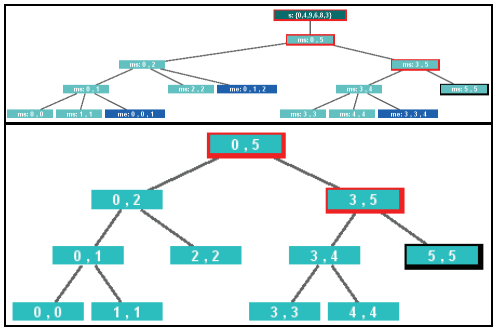

Figura 11. Árbol de recursión de ordenación por mezcla mostrando las llamadas de tres o un método

Conectar – espacio. "Mostrar elementos relacionados". Esta categoría permite resaltar las

relaciones entre elementos de un programa o mostrar elementos ocultos pero relacionados. Algunas conexiones se encuentran incluidas en la propia representación gráfica, pero también pueden resaltarse otras menos evidentes.

Las múltiples vistas que soporta SRec no son independientes, sino que están coordinadas. Esta conexión se consigue de dos formas. Primero, mediante la sincronización de una animación de la información mostrada en las distintas vistas. Segundo, mediante la compartición de convenios visuales en las diferentes representaciones de los mismos objetos. En SRec, comparten convenios de color y de orientación espacial.

Conectar – tiempo. "Mostrar instantes relacionados". Es similar, pero con respecto a instantes de una ejecución.

Aunque no está implementado actualmente en SRec, podrían conectarse los estados de una estructura de datos que representan el valor de entrada y de salida de una misma llamada de un algoritmo de divide y vencerás. Esta interacción podría efectuarse sobre la visualización mostrada en la parte izquierda de la Figura 4.

4. Conclusiones y trabajos futuros

Hemos identificado los problemas que presenta la taxonomía de la interacción en visualización de la información cuando se aplica a visualización del *software*. Hemos modificado la propuesta de Yi *et al.* mediante una definición más precisa de las categorías y la distinción de las dimensiones espacio y tiempo. Hemos aplicado la taxonomía resultante a los sistemas desarrollados, resultando muy clarificadora.

Ya hemos comentado en algunas de las categorías que el tiempo puede representarse de varias formas [4]. Un trabajo futuro será analizar las consecuencias de las diversas representaciones del tiempo sobre la interacción.

También conviene revisar otras taxonomías de interacción en visualización de la información. Aunque en principio son menos adecuadas, pueden contener aspectos interesantes.

Agradecimientos

Este trabajo se ha financiado con el proyecto TIN2008-04103/TSI del MICINN.

Referencias

[1] Almeida Martínez, F.; Urquiza Fuentes, J; Velázquez Iturbide, J.Á. "Visualization of syntax trees for language processing courses". *J. Universal Computer Science*, 15(7):1546-1561, 2009.

[2] Card, S. K.; Mackinlay, J. D.; Shneiderman, B. (eds.) *Readings in Information Visualization*. Morgan Kaufmann, 1999.

[3] Diehl, S. *Software Visualization*. Springer, 2007.

[4] Gómez Henríquez, L. *Sistematización y uso de las técnicas de visualización de programas concurrentes*. Tesis doctoral, Universidad Politécnica de Madrid, 1995.

[5] Hundhausen, C.; Douglas, S.; Stasko, J. "A meta-study of algorithm visualization effectiveness". *J. Visual Languages and Computing*, 13(3):259-290, 2002.

[6] Karavirta, V.; Korhonen, A.; Malmi, L.; Naps, T. "A comprehensive taxonomy of algorithm animation languages". *J. Visual Languages and Computing*, 21(1):1-22, 2010.

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

[8] Price, B.; Baecker, R.; Small, I. "A principled taxonomy of software visualization". *J. Visual Languages and Computing*, 4(3):211-266, 1993.

[9] Roman, G.-C.; Cox, K. C. "A taxonomy of program visualization systems". *Computer*, 26(12):11-24, 1993.

[10] Stasko, J.; Domingue, J.; Brown, M.; Price, B. (eds.) *Software Visualization*. The MIT Press, 1998.

[11] Urquiza Fuentes, J; Velázquez Iturbide, J.Á. "A survey of successful evaluations of program visualization and algorithm animation systems". *ACM Trans. Computing Education*, 9(2):artículo 9, 2009.

[12] Velázquez Iturbide, J.Á. "Principled design of logical fisheye views of functional expressions". *ACM SIGPLAN Notices*, 41(8): 34-43, 2006.

[13] Velázquez Iturbide, J.Á.; Pareja Flores, C.; Urquiza Fuentes, J. "An approach to effortless construction of program animations". *Computers & Education*, 50(1):179-192, 2008.

[14] Velázquez Iturbide, J.Á.; Pérez Carrasco, A. "Active learning of greedy algorithms by means of interactive experimentation". En *Proc. 14th Annual Conf. Innovation and Technology in Computer Science Education*, pp. 119-123, 2009.

[15] Velázquez Iturbide, J.Á.; Pérez Carrasco, A. "InfoVis interaction techniques in animation of recursive programs". *Algorithms* 2010(3): 76-91, 2010.

[16] Velázquez Iturbide, J.Á.; Pérez Carrasco, A.; Urquiza Fuentes, J. "SRec: An animation system of recursion for algorithm courses". En *Proc. 13rd Annual Conf. Innovation and Technology in Computer Science Education*, pp. 225-229, 2008.

[17] Yi, J.S.; Kang, Y.a.; Stasko, J.T.; Jacko, J.A. "Toward a deeper understanding of the role of interaction in information visualization", *IEEE Trans. Visualization and Computer Graphics* 13(6):1.224-1.231, 2007.