Visualización de la colaboración en la evolución de un ítem de software y la estructura de las *baselines*[*]

Roberto Therón* Antonio González** Francisco J. García***

Departamento de Informática y Automática

Universidad de Salamanca

Facultad de Ciencias. Plaza de los Caídos, s/n

37008 Salamanca, España

theron@usal.es* agtorres@usal.es** fgarcia@usal.es***

Resumen

En este trabajo proponemos una visualización para el historial de la evolución de un ítem de software y la estructura de las *baselines*, utilizando como fuente de información los repositorios de datos de las herramientas de Administración de la Configuración de Software (SCM). La propuesta se apoya en múltiples técnicas de interacción y visualización para proporcionar vistas enlazadas entre la representación del árbol de revisiones y la estructura de las *baselines*, mantener ubicado al usuario en el contexto mediante una línea temporal, ver y comparar las estructuras de las *baselines* y presentar las relaciones de modificación sobre los elementos de la estructura de una *baseline*.

1. Introducción

El desarrollo de un producto de software es un complejo proceso que requiere una estrecha colaboración, comunicación y coordinación entre diseñadores, desarrolladores y administradores de equipos de trabajo distribuidos geográficamente. Por lo que es de gran importancia contar con información precisa sobre las tareas que ejecuta cada participante y la colaboración que tiene lugar entre ellos.

El proceso de administración de la configuración del software y la documentación es definido por el estándar IEEE 828-1990[3] como "un sistema que proporciona los métodos y herramientas para identificar y controlar el software durante su desarrollo y uso. En donde las actividades de este proceso incluyen la identificación y establecimiento de puntos de referencia; la revisión, aprobación y control de cambios; el seguimiento y reporte de esos cambios; las auditorías y revisiones de la evolución de un producto software". De aquí la importancia que tienen los repositorios de las herramientas SCM[1] para la extracción de información que permita identificar el establecimiento de *baselines* y revisiones, fechas y horas, para el seguimiento de los cambios que sufre un sistema o ítem de software. Sin embargo, a pesar de la riqueza de esta fuente de datos las herramientas SCM todavía no incorporan mecanismos para representar como ocurren las contribuciones y la colaboración entre los miembros de un proyecto de software.

Este trabajo se enfoca en la visualización de la evolución de los ítems de software, la colaboración que tiene lugar durante su desarrollo y la representación de las estructuras de las *baselines*, haciendo uso de los repositorios de las herramientas SCM. Para ello nos apoyamos en el uso de múltiples técnicas de inte-

[*]Queremos agradecer la ayuda del Ministerio de Educación y Ciencia, a través del Programa Nacional de Tecnologías de Servicios de la Sociedad de la Infomación (proyecto ref. TSI2005-00960 y del Programa Nacional de Tecnologías Informáticas (proyecto ref. TIN2006-06313)

[1]Software Configuration Management tools

racción y representación de información; vistas enlazadas, utilización del *zoom*, contexto sensible al ratón, animaciones, uso de estructuras matriciales para mostrar la colaboración entre desarrolladores, una línea temporal, una estructura radial, y una representación polifocal, entre otras más.

En la sección 2 de este documento realizamos una revisión bibliográfica sobre la visualización de la evolución de versiones de software, en la sección 3 presentamos nuestra propuesta de visualización, en la sección 4 efectuamos una comparación entre nuestro diseño y otras propuestas. Finalmente, en la sección 5 hacemos un resumen del trabajo y discutimos las principales conclusiones.

2. Trabajos relacionados

Una considerable cantidad de trabajos han sido dedicados al estudio de la visualización de software e información. Gracanin [9] sostiene que la visualización de software es "una disciplina que hace uso de formas imaginativas que proporcionan el detalle y entendimiento suficiente de un sistema reduciendo la complejidad aparente de éste".

Durante el diseño de la representación se deben identificar las tareas que serán realizadas por la visualización, el alcance y contenido de ésta, quienes serán la audiencia, qué datos serán representados y cómo, qué medios se utilizaran para la representación y las técnicas de visualización e interacción.

En este artículo nos concentramos en la evolución de ítems individuales y la colaboración durante su desarrollo, y dedicamos esta sección a revisar algunas ideas útiles que han sido aplicadas a la representación de repositorios de las herramientas SCM.

Xie et al. [29] lista una serie de preguntas para guiar el diseño de visualizaciones de los repositorios de las herramientas SCM; para nuestros propósitos es relevante determinar los autores que trabajan en el mismo ítem, cuando fue realizada una modificación y cuántos programadores han trabajado en una versión del sistema. De forma adicional, Eick et al. [7] manifiesta de forma acertada que un problema fundamental en la visualización de los cambios del software es la selección de representaciones o metáforas visuales efectivas, revisarlas en detalle y combinarlas para obtener diferentes perspectivas de los datos, por ejemplo; por desarrollador, por estadísticas sobre los cambios, tamaño de los cambios y actividades efectuadas por los desarrolladores, entre otras más.

Voinea y Telea [24] respaldan la idea de que los repositorios de administración de la configuración son de gran valor para contabilizar información del proyecto, realizar auditorías y entender la evolución de los proyectos de software. Nosotros apoyamos ese criterio y consideramos que el diseño efectivo de los repositorios puede proporcionar información acerca del desarrollo que no es posible obtener de otras fuentes y que mediante una visualización adecuada es posible obtener un gran número de detalles sobre lo que sucede durante el proyecto. Los mismos autores proponen dos visualizaciones en [25] y [26]. Dichas propuestas demuestran que el uso adecuado de visualizaciones en 2D en conjunto con colores y texturas contribuyen al desarrollo de poderosas soluciones de visualización multidimensional.

Gall et. al. [8] desarrollaron un enfoque interesante que utiliza representaciones en 3D con codificación de colores aplicada a los cambios en la estructura y atributos del ítem. Los atributos bajo consideración son el número de revisión, tamaño del ítem y complejidad. Este enfoque visualiza las versiones del sistema como una sucesión de representaciones en 2D, en donde cada una corresponde a una versión y utiliza colores diferentes por atributo.

En lo que respecta a la representación de espacios temporales, Morris et al. [17] visualizan jerarquías temporales de documentos en una línea de tiempo de acuerdo con la estructura jerárquica producida en la etapa de *clustering*. Por otro lado, Card et al. [4] desarrollaron una visualización para explorar jerarquías que cambian con el tiempo, soportando búsquedas, navegación a través de la estructura y filtrado de resultados mediante un *time slider*. Therón [23] propone un metáfora del tipo *tree-ring* para representar estruc-

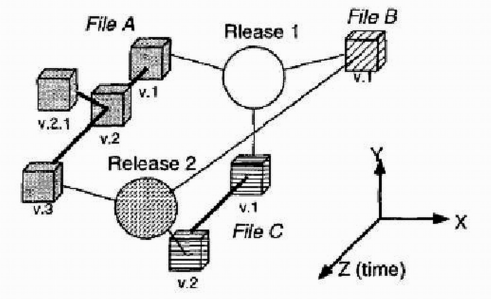

Figura 1: Visualización del historial de versiones de software utilizando 3D.

turas basadas en jerarquías temporales con el fin de facilitar la navegación y descubrir relaciones en la historia de los lenguajes de programación. Mientras, Kumar y Garland [14] proponen una solución para la visualización de grafos que varían en el tiempo, permitiendo que los usuarios interactúen con la representación y puedan deslizarse a diferentes periodos para explorar un grafo o descubrir tendencias.

La propuesta de Lanza en [16] visualiza la evolución de atributos de software mediante una matriz de evolución con cajas rectangulares de tamaño variable en cada celda; el ancho de las cajas representan el número de métodos y el alto el número de variables en la clase. Esta representación es poderosa y puede ser mejorada tomando prestadas algunas ideas sobre colores y texturas de [8].

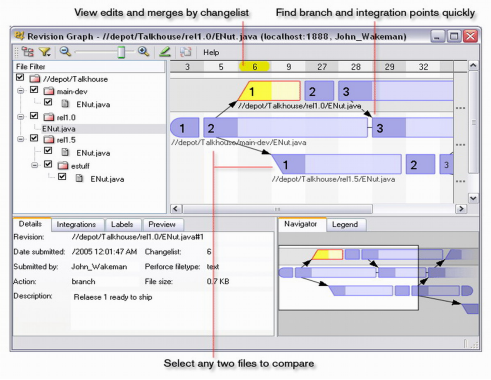

Figura 2: Visualización realizada por Perforce.

En este punto, es relevante hacer referencia al trabajo desarrollado por Koike en [12, 13] y que describe VRCS, una representación que muestra la evolución de los ítems de software con base en la información de los repositorios de información de las herramientas SCM. En esta visualización cada ítem de software es representado utilizando 2D y el repositorio en general mediante 3D, como se ilustra en la figura 1. Esta visualización será analizada más adelante, en la presentación del caso de estudio.

Por otro lado, Perforce (http://www.perforce.com/) es una herramienta SCM que incluye un módulo de visualización (ver figura 2), cuyas representaciones son en 2D y utiliza grafos para mostrar las relaciones entre las *baselines*, ramas y revisiones. Esta representación ofrece una vista del tipo *overview + detail* en lugar de una vista del tipo foco + contexto [19] que hubiera resultado en un enfoque más conveniente. En la sección del caso de estudio discutiremos con más detalle esta propuesta. Los repositorios de las herramientas SCM son estructuras jerárquicas de directorios y archivos o sus correspondientes representaciones almacenadas en una base de datos. Existe una importante cantidad de trabajos que se enfocan en la visualización de estructuras jerárquicas. Una de las representaciones más conocidas es el treemap [11], del cual existen diversas variantes [2, 28, 10, 21].

Otras representaciones jerárquicas bastante conocidas son la hiperbólica [15], Cone Trees [20] (en [6] se hace un análisis de rendimiento de ésta), así como las circulares y semicirculares [1, 22, 10, 18, 30]

Las visualizaciones jerárquicas son de especial relevancia en lo que respecta a la visualización de la estructura de las *baselines*, en nuestro caso hemos seleccionado el uso de un grafo radial para su representación.

3. Evolución ítems de software y estructura de las *baselines*

En esta sección presentamos nuestra propuesta para representar la evolución de un ítem

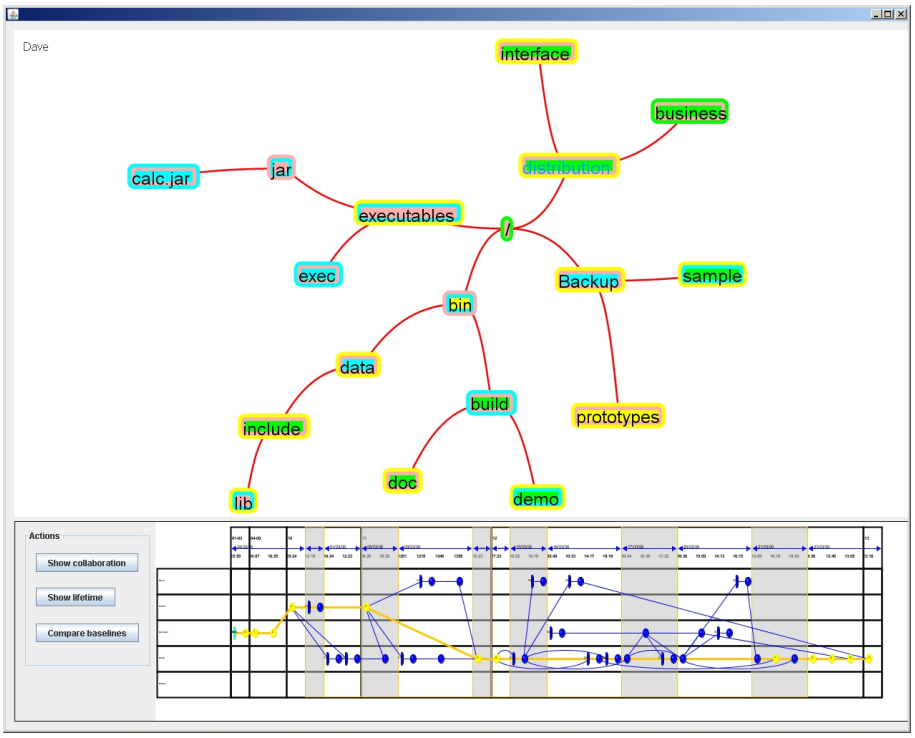

Figura 3: Visualización de la colaboración en el desarrollo de ítems de software y la estructura de *baselines*.

de software y la estructura de las *baselines*, utilizando como fuente de información los repositorios de las herramientas SCM. Para ello nos apoyamos en dos representaciones, el *Revision Tree* y el *Baseline Navigator*, así como en múltiples técnicas de interacción. La figura 3 muestra la vista principal de la visualización, en ella se puede distinguir la estructura de una *baseline* y la colaboración en la modificación de cada elemento mediante el uso de anillos codificados con colores, donde cada color se corresponde con un desarrollador. Cuando el ratón pasa por encima de los anillos se presenta en una etiqueta el nombre del programador que hizo el cambio correspondiente. Un trabajo importante que hace un uso similar de anillos de colores es descrito en [5]. Adicionalmente, la representación cambia el color de la letra de un elemento cuando este es seleccionado, como se puede notar en el caso del elemento distribución.

Antes de continuar, es importante resaltar que una *baseline* es una referencia al estatus de un ítem particular en un momento determinado, por lo que su estructura corresponde a la estructura del proyecto en un determinado instante.Finalmente, la estructura de las *baselines* no es común a todos los ítems de software en el mismo número de *baseline*.

El *Revision Tree* visualiza las contribuciones de los programadores en diferentes revisiones, *baselines* y largos periodos de tiempo, en el mismo ítem, prestando especial consideración al factor temporal. La visualización de la evolución de un ítem presenta múltiples retos; la representación de grandes árboles de revisión donde las *baselines* tienen muchas ramas y cada rama un gran número de revisiones; la navegación a través del árbol de revisiones ofreciendo una vista del tipo foco + contexto,

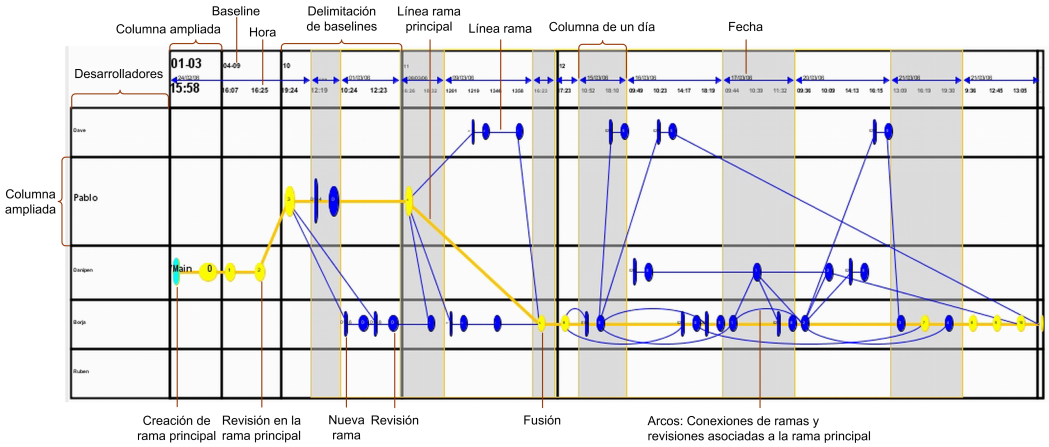

Figura 4: Visualización de la colaboración en el desarrollo de ítems de software.

soporte a la interacción para permitir la revisión de más de una *baseline* a la vez, mostrar la colaboración de los desarrolladores en cada *baseline*, y relacionar la información con la línea de tiempo.

El cuadro 1 y la figura 4 muestran y describen los elementos gráficos utilizados en el *Revision Tree*.

El *Revision Tree* utiliza columnas de ancho variable para acomodar las revisiones en cada *baseline*, y filas con distribución uniforme para los programadores, además permite expander filas y columnas con un enfoque polifocal para apreciar la relación jerárquica entre elementos. La primera fila corresponde a la línea temporal cuyos componentes son la numeración de las *baselines*, fecha y hora de la creación de una revisión o rama y la representación del área correspondiente a un día.

Aunque de momento no ha sido implementado, la propuesta soporta interacción adicional para proporcionar detalles en árboles de revisión complejos; el usuario puede seleccionar la rama principal o una rama regular y se resalta el camino de dicha rama con todas sus asociaciones y también es posible seleccionar un nodo para resaltar sus conexiones. Nuestra propuesta es que el usuario pueda obtener la información que requiere mediante el uso de interacción.

Elemento Visual	**Descripción**
Desarrolladores	Nombre del desarrollador.
Baseline	Número de *baseline*.
Fecha	Fecha de creación de las ramas, *baselines* o revisiones.
Día	Línea azul con flechas en los extremos; día de actividad en la creación de las ramas, *baselines* y revisiones.
Hora	Hora de creación de una nueva rama o revisión.
Creación rama principal	Óvalo de color fucsia.
Nueva rama	Óvalo alargado de color azul.
Rama principal	Línea color naranja.
Arcos azules	Conecta las revisiones de las ramas en el mismo workspace de la rama principal.
Revisiones rama principal	Nodos de color amarillo.
Rama	Líneas de color azul que conectan los elementos de las ramas y éstas con la principal.
Revisión	Nodos de color azul.
Fusión	Flechas entrantes en la rama principal.

Cuadro 1: Variables representadas por el *Revision Tree*.

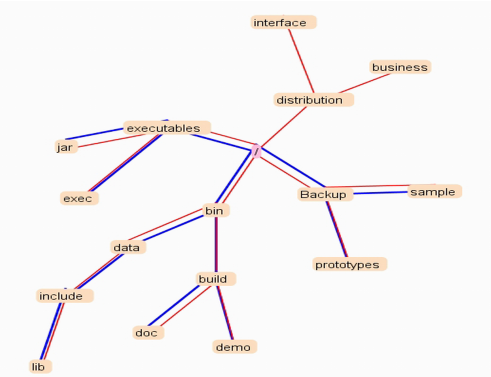

Figura 5: Comparación estructura de *baselines*.

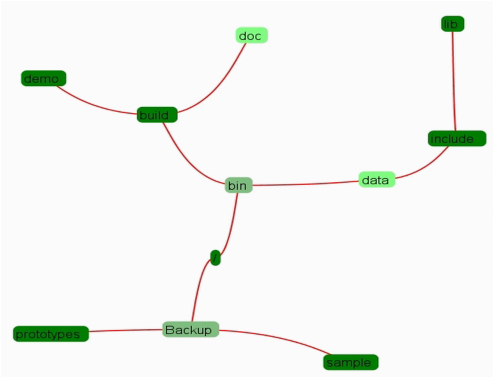

Figura 6: Antigüedad modificaciones de nodos.

Con el *Revision Tree* es posible obtener rápidamente datos de relevancia, como dar seguimiento a las contribuciones al desarrollo de un ítem de software, determinar quiénes han estado involucrados en el proceso y proporcionar información de utilidad a los administradores de proyectos; si alguien está de baja es posible descubrir si las últimas revisiones de ese programador fueron fusionadas.

También es posible obtener información sobre los periodos con más actividad en el ítem y reconocer cuando ha entrado en una fase de estabilidad porque no sufre cambios frecuentes. En realidad, es posible obtener más información revisando cuidadosamente cada detalle de la visualización y especialmente si se utiliza un conjunto de datos real de gran tamaño.

Mediante la interacción del *Revision Tree* y el *Baseline Navigator* es posible examinar la estructura de una *baseline*, comparar la estructura de dos *baselines* y determinar cuáles elementos han sido modificados recientemente.

Al comparar la estructura de dos *baselines*, la sección común de éstas aparece resaltada por dos aristas, la parte de la sección que corresponde a la adición de elementos utiliza aristas de color rojo y la sección que ha sido eliminada aristas de color verde. La figura 5 es un ejemplo de la comparación de dos *baselines*.

Por otro lado, la figura 6 hace referencia a la antigüedad de las modificaciones de los nodos utilizando una escala de tonos verdes, en donde el tono más claro hacer referencia a la modificación más reciente.

El *Baseline Navigator*, soporta técnicas de interacción y animación que permiten ubicar un nodo seleccionado en el centro de la visualización, hacer *zoom* en cualquier punto de la estructura y arrastrar elementos. Finalmente, es importante mencionar que su desarrollo se realizó con Prefuse, al cual se le realizaron cambios en algunas clases y adicionaron otras para lograr la representación deseada.

4. Caso de estudio: Ventajas frente a los árboles de revisión en 3D

El desarrollo de representaciones visuales en 3D ha sido muy popular en los últimos años. Nosotros utilizamos una representación en 2D del *Revision Tree* para visualizar árboles de revisión complejos y comparamos los resultados con los producidos por la herramienta de visualización en 3D de PlasticSCM, producida por Códice Software, VRCS y Perforce. En esta comparación no consideramos el *Baseline Navigator* debido a que las otras propuestas no visualizan la estructura de las *baselines*.

La figura 7 muestra el árbol de evolución en 3D para el mismo ejemplo de la figura 4.

Aunque el *Revision Tree* se aprecia mejor en pantallas de gran tamaño, aún es espacios reducidos ayuda a obtener una idea general sobre la evolución de un ítem. Lo anterior se

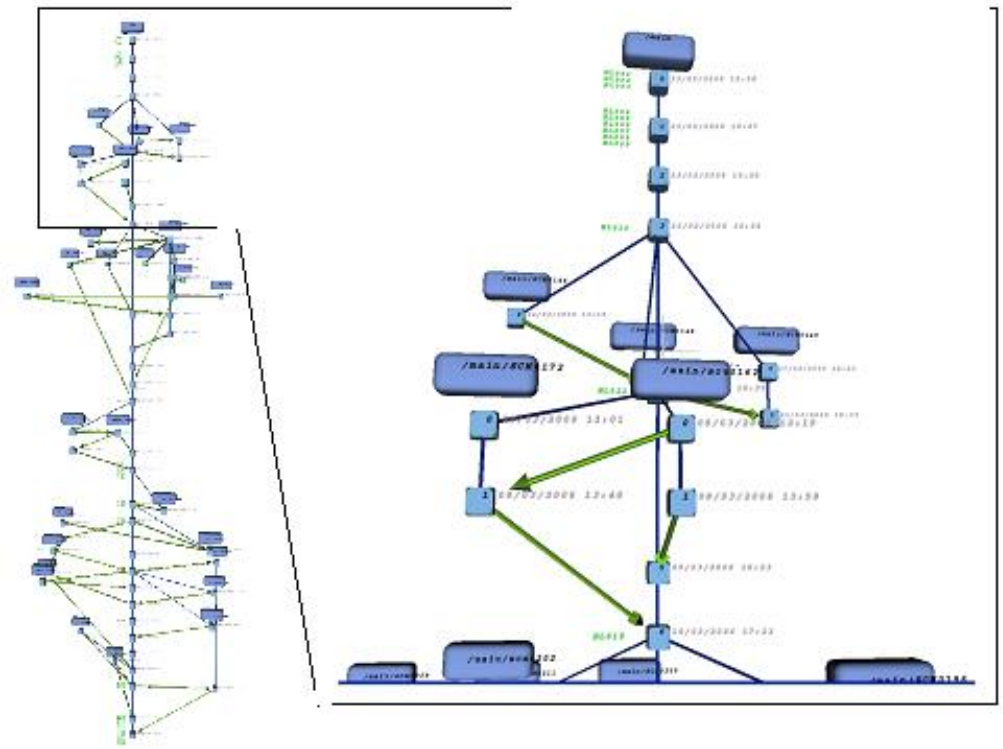

Figura 7: Ejemplo del árbol producido por PlasticSCM y un *zoom* en una área seleccionada.

ilustra en la figura 8, que expande en un 80 % del área de visualización la fila correspondiente a Borja.

Hasta este punto, el trabajo realizado por Ware [27] ha sido de gran ayuda: en él se analiza el uso de 3D en la visualización de información y propone el uso de una actitud de 2 1/2D al diseñar representaciones; sugiere el uso consciente de visualizaciones en 3D en combinación con 2D para producir mejores soluciones.

Con respecto a VRCS, figura 1, nuestra principal preocupación es la ausencia de una vista del tipo foco + contexto, la navegación a través de la estructura del árbol, posibles oclusiones y el comportamiento de ésta al representar estructuras complejas que demanden procesamiento y memoria de forma intensiva. Como ya hemos discutido, la visualización de largas historias de revisiones para un ítem utilizando un árbol de versiones en 3D tiene algunas limitaciones. Por lo que la visualización de grandes repositorios conteniendo muchos ítems, con numerosas *baselines* y revisiones podría resultar en una visualización de difícil navegación y probablemente no proporcione rápidamente la información requerida por el usuario.

En el caso de la representación producida por Perforce (figura 2), ofrece una vista del tipo *overview + detail*, por lo que se pierde gran cantidad de espacio utilizable; muestra información sobre las ramas y fusiones, y es posible obtener la fecha y hora de creación de las revisiones haciendo clic en los nodos y revisando la información en el tab *Details* que se

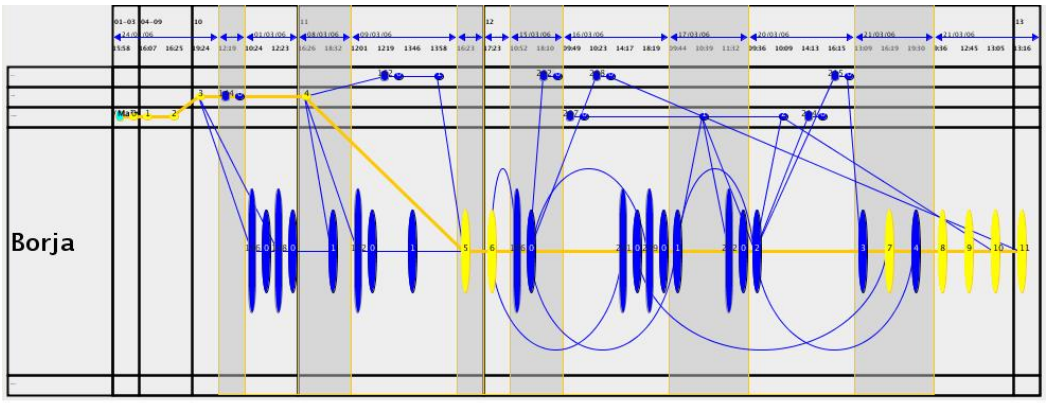

Figura 8: *Revision Tree*: Columna expandida tanto a lo ancho como a lo alto.

encuentra en el panel a la izquierda de la pantalla. Sin embargo, la representación no proporciona información sobre los programadores que están contribuyendo al desarrollo de un ítem, por cuánto tiempo han estado trabajando en él o ver la línea temporal de un vistazo. Adicionalmente, esta visualización es estática y no ofrece opciones de interacción.

Finalmente, en el cuadro 2 presentamos una comparación entre los enfoques de representación analizados, utilizando como base una serie de preguntas.

5. Conclusiones

La representación propuesta visualiza el historial de evolución de un ítem de software y la estructura de las *baselines*. En el caso del *Revision Tree*, este proporciona una vista del tipo foco + contexto, una estructura matricial y una línea de tiempo para guiar y posicionar a los usuarios en el espacio temporal. Por otro lado el *Baseline Navigator*, representa la estructura de las *baselines* y ofrece información sobre cuáles elementos han sido modificados de forma más reciente, quienes han realizado las modificaciones y permite comparar la estruc-

Preguntas	**Pl**[1]	**VR**[2]	**Pe**[3]	**RT**[4]
¿Cuántos programadores han participado en el desarrollo del ítem?			X	X
¿Quiénes han contribuido con la evolución del ítem?				X
¿Quién es el programador que más ha contribuido en el desarrollo del ítem?				X
¿Cuántas *baselines* constituyen la evolución del ítem?				X
¿Existen revisiones del ítem sin fusionar?	X	X		X
¿Cuánto tiempo ha durado el desarrollo del ítem?			X	X
¿Cuál *baseline* tiene más ramas y revisiones?			X	X
¿Cuál rama tiene más revisiones?	X	X	X	X
¿En qué periodo el ítem no tiene actividad?			X	X
¿Existe algún periodo con cambios bruscos de actividad?				X

Cuadro 2: Comparación de representaciones de los árboles de revisión

[1] PlasticSCM
[2] VRCS
[3] Perforce
[4] *Revision Tree*

tura de dos *baselines*. Adicionalmente, consideramos el uso de gran cantidad de técnicas de interacción para hacer posible la integración de múltiples vistas y la navegación a través de la representación.

Con esta representación el usuario obtiene respuestas sobre lo que ha ocurrido en la evolución de un ítem y la estructura de las *baselines*, con el beneficio de que el equipo de desarrollo se mantiene al tanto de quienes han trabajado en las diferentes revisiones y *baselines*.

Con el apoyo de las técnicas de interacción y la visibilidad permanente de la visualización, los usuarios pueden revisar información sobre las revisiones, *baselines* y la estructura de las *baselines*; en un periodo corto de tiempo, sin que ocurra oclusión u ocultamiento de información.

Por otro lado, la línea de tiempo es una representación clara que muestra el intervalo de tiempo desde que un ítem fue creado, soporta comparaciones temporales y proporciona detalles sobre la concurrencia de los desarrolladores. Además, de que la interactividad permite filtrar contenidos para enfocarse en una o múltiples áreas de interés; en síntesis la visualización en 2D ofrece una representación clara y funcional.

Considerando lo anterior, esta propuesta proporciona suficiente evidencia para afirmar que la representación de la evolución y la colaboración en el desarrollo de ítems de software utilizando una representación en 2D, apoyada en el uso de múltiples posibilidades de interacción puede resultar en una poderosa solución para visualizar datos multidimensionales.

Referencias

[1] Andrews, K., *Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs*, Late Breaking Hot Topic Paper, IEEE Symposium on Information Visualization (InfoVis 98), 1998.

[2] Balzer, M., *Voronoi treemaps for the visualization of software metrics*, Proceedings of the 2005 ACM symposium on Software visualization, ACM Press, New York, USA, 2005.

[3] Berlack, H.R., *IEEE standard for software configuration manager plans*, IEEE, 1990.

[4] Card, S., *TimeTree: Exploring Time Changing Hierarchies*, IEEE Symposium on Visual Analytics Science and Technology, 2006.

[5] Chaomei, C., *Detecting and visualizing emerging trends and transient patterns in scientific literature*, Journal of the American Society for Information Science and Technology, 1(57), 2006.

[6] Cockburn, A., *An evaluation of cone trees*, Proceedings of the 2000 British Computer Society Conference on HumanComputer Interaction, 2000.

[7] Eick, S.,*Visualizing software changes*, IEEE Trans. Softw. Eng., 28(4), 2002.

[8] Gall, H., *Visualizing software release histories: The use of color and third dimension*, Proceedings of the IEEE International Conference on Software Maintenance, IEEE Computer Society, Washington, DC, USA, 1999.

[9] Gracanin, D., *Software visualization*, Innovations in Systems and Software Engineering: A NASA Journal, vol. 1, 2005.

[10] Goodrich, M., *RINGS: A Technique for Visualization of Large Hierarchies*, Graph Drawing 2002, Springer-Verlag, April 2002.

[11] Johnson, B., *Tree-maps: a space-filling approach to the visualization of hierarchical information structures*,Proceedings of the 2nd conference on Visualization 91,IEEE Computer Society Press, Los Alamitos, USA, 1991.

[12] Koike, H., *The role of another spatial dimension in software visualization*, ACM Trans. on Information Systems, 11(3), 1993.

[13] Koike, H., *Vrcs: Integrating version control and module management using interactive 3d graphics*, Proceedings of the 1997 IEEE Symposium on Visual Languages (VL 97), IEEE Computer Society, Washington, DC, USA, 1997.

[14] Kumar, G., *Visual Exploration of Complex Time-Varying Graphs*, IEEE Transactions on Visualization and Computer Graphics, Vol. 12, Num. 5, 2006.

[15] Lamping. J., *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*, Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press/Addison-Wesley Publishing Co, New York, USA, 1995.

[16] Lanza, M., *The evolution matrix: recovering software evolution using software visualization techniques*, Proceedings of the 4th International Workshop on Principles of Software Evolution, ACM Press, New York, NY, USA, 2001.

[17] Morris, S., *Time line visualization of research fronts*, Journal of the American Society for Information Science and Technology, vol. 54, 2003.

[18] Pavlo, A., *A parent-centered radial layout algorithm for interactive graph visualization and animation*, ArXiv Computer Science e-prints, jun 2006.

[19] Rao, R., *The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information*, Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, New York, NY, USA, 1994.

[20] Robertson, G., *Cone trees: animated 3d visualizations of hierarchical information*, Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, New York, NY, USA, 1991.

[21] Shneiderman, B., *Treemaps for space-constrained visualization of hierarchies*, Human-Computer Interaction Lab / University of Maryland, 28 de abril 2006. Consultado el 15 de octubre del 2006.

[22] Stasko, S., *Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations*, Proceedings of the IEEE Symposium on Information Vizualization 2000, IEEE Computer Society, Washington, DC, USA, 2000.

[23] Theron, R., *Hierarchical-temporal Data Visualization using a Tree-ring Metaphor*, Lecture Notes in Computer Science. Smart Graphics, Springer-Verlag, vol. 4073, Germany, 2006.

[24] Voinea, L., *An open framework for cvs repository querying, analysis and visualization*, Proceedings of the 2006 international workshop on Mining software repositories, ACM Press, New York, NY, USA, 2006.

[25] Voinea, L., *Multiscale and multivariate visualizations of software evolution*, SOFTVIS 2006, Association for Computing Machinery Inc., 2006.

[26] Voinea, L., *Mining software repositories with cvsgrab*, Proceedings of the 2006 international workshop on Mining software repositories, ACM Press, New York, NY, USA, 2006.

[27] Ware, C., *Designing with a 2 1/2D Attitude*, Information Design Journal, 10(3), 2001.

[28] Wetzel, K., *Pebbles: Using circular treemaps to visualize disk usage*, http://www.SourceForge.net, Setiembre 2004.

[29] Xie, X., *Visualization of cvs repository information*, Proceedings of the 13th Working Conference on Reverse Engineering (WCRE 2006), IEEE Computer Society, Washington, DC, USA, 2006.

[30] Yang, J., *Interring: a visual interface for navigating and manipulating hierarchies*, Information Visualization, 2(1), 2003.