GridMaster para entornos web: una visualización analítica de la evolución del software

Orlando Vega-Esquivel[1], Hugo Adams-Barrantes[1], Ana Chévez-Mora[1], and Antonio González-Torres[1]

Escuela de Ingeniería,
Universidad Latinoamericana de Ciencia y Tecnología,
ULACIT, Urbanización Tournón, 10235-1000
San José, Costa Rica
ovegae130, hadamsb275, achevezm587, agonzalez@ulacit.ac.cr
http://www.ulacit.ac.cr

Resumen Gracias al avance tecnológico, hoy en día es más común encontrar equipos de desarrolladores de software distribuidos en diferentes locaciones geográficas, utilizando entornos integrados de desarrollo en la nube, tendencia que es más evidente en las comunidades de desarrollo de código libre. La analítica visual es una herramienta que facilita el trabajo de mantenimiento del software en estos ambientes colaborativos, pues se debe hacer frente a constantes cambios a medida que el software crece y se vuelve más complejo. En el presente documento se tratará la factibilidad tecnológica y los beneficios que brindaría la utilización de la visualización analítica en el seguimiento de la evolución del software. Este argumento se fundamenta mediante la implementación de un prototipo web para la visualización conocida como GridMaster, así como de un eventual reconocimiento de las capacidades de inferencia que se derivan de su uso.

Keywords: Analítica visual, Visualización analítica, Evolución de software, Visualización de software, Ambientes integrados de desarrollo en la nube, GridMaster.

1 Introducción

Los proyectos de software son entornos en constante evolución debido a factores como cambios en los requerimientos, correcciones de errores e implementación de nuevas funcionalidades. A medida que pasa el tiempo, el software crece y se vuelve más complejo, de manera que se hace necesario dedicar mucho tiempo a su mantenimiento (D'Ambros & Lanza, 2009). Bajo este contexto, es importante contar con herramientas que permitan dar seguimiento a la evolución de los componentes de software y a las relaciones que existan entre los desarrolladores participantes.

La analítica visual puede utilizarse para facilitar un mejor entendimiento de la información al reducir la sobrecarga cognitiva (Karanam & Akepogu, 2011)

con el uso de herramientas de software que maximicen la capacidad humana para percibir, comprender y razonar datos complejos y situaciones dinámicas (Thomas & Cook, 2005). Por ende, la analítica visual ayudaría a optimizar la interpretación efectiva de la evolución del software, la cual es necesaria para dar un adecuado seguimiento y control a los proyectos de desarrollo.

El presente documento se basa en el diseño previo de una visualización analítica de la evolución del software llamada *GridMaster*, el cual se detalla en el artículo *Maleku: an evolutionary visual software analytics tool for providing insights into software evolution* de Antonio González Torres et al. (González, Therón, García, Wermelinger, & Yu, 2011). Dicha visualización traza en forma matricial el historial de cambios de la jerarquía de componentes de código de un proyecto de software a lo largo de su desarrollo. Además, permite identificar por medio de colores los desarrolladores responsables de los cambios, estimar la tasa de cambios asociada y determinar las relaciones entre los desarrolladores.

En el desarrollo de este documento se hará alusión a la factibilidad tecnológica para la implementación de la visualización analítica GridMaster en entornos integrados de desarrollo (IDE) en la nube, así como a los posibles beneficios que con ello se obtendrían. Esta implementación agrega la complejidad de las consideraciones que deben tenerse presentes en el uso de tecnologías completamente web.

Lo que resta del artículo se compone de las siguientes secciones:

- Una breve reseña de las más recientes tendencias con respecto a los entornos de desarrollo en la nube y los proyectos de código libre.
- Los conceptos generales asociados a la analítica visual, y la justificación de cómo esta rama de la computación se presta para ser una herramienta de apoyo en el ciclo de vida de los proyectos software.
- Una descripción de la visualización GridMaster y sus principales casos de aplicación.
- Una descripción de la propuesta de implementación de la visualización GridMaster para IDE de nube.
- Un análisis de resultados de la implementación de un prototipo funcional de la visualización GridMaster para entornos web.
- Las conclusiones del presente trabajo, derivadas de los contenidos anteriores.
- Recomendaciones para futuras investigaciones asociadas a la temática de este documento.

2 Entornos de desarrollo en la nube y proyectos de código libre

En los últimos años, la computación en la nube ha tenido un aumento considerable, tanto así que los productos de software tradicionales que deben ser instalados en cada máquina, han ido cambiándose gradualmente al denominado software como servicio (SaaS) (Albaroodi, Manickam, & Aboalmaaly, 2013). Este comportamiento también se ha dado con las herramientas empleadas para el desarrollo de software.

Los proyectos de desarrollo de software se han convertido en demandantes de entornos colaborativos, en los que los desarrolladores puedan trabajar de manera conjunta para construir soluciones de software. Ahora bien, gracias a los avances tecnológicos como el de la computación en la nube, es cada vez más común encontrar equipos de desarrolladores distribuidos en diferentes locaciones geográficas, los cuales utilizan ambientes de desarrollo colaborativo a través de Internet. En este tipo de escenarios, se hace indispensable una adecuada coordinación del trabajo durante todo el ciclo de vida del desarrollo, de tal manera que se logren evitar cantidades significativas de errores y duplicidad de esfuerzos (Ferzund, Yasrab, & Razzaq, 2014).

Algunas de las principales características de la computación en la nube son la accesibilidad, el apoyo a la innovación y la reducción de costos (Albaroodi et al., 2013); la tendencia de migrar los desarrollos a entornos de nube se ha evidenciado con mayor fuerza en las comunidades de desarrollo de código libre, donde tales características son de gran relevancia.

A pesar de que la utilización de entornos de desarrollo en la nube no ha tenido gran difusión hasta el momento, ha aumentado la cantidad de herramientas de este tipo, como también han aumentado las capacidades de las herramientas ya existentes; por lo cual, debe considerarse que es un nicho de mercado creciente.

3 Analítica visual como herramienta de apoyo en el ciclo de vida del software

3.1 Conceptos generales

Existen varios conceptos que deben aclararse para entender el potencial inherente en la aplicación de la analítica visual y su utilización en la ingeniería de software; a continuación se explican algunos de ellos.

Análisis: de acuerdo con Thomas y Cook, el concepto de análisis puede ser visto como un arte y como una ciencia, cuyo objetivo principal es realizar juicios sobre un tema o una cuestión relacionada con un problema de mayor tamaño. Estos mismos autores afirman que los analistas a menudo deben llegar a sus juicios bajo una presión significativa de tiempo, así como con información limitada o conflictiva, por lo que sus juicios reflejan necesariamente su mejor comprensión de una situación, con los supuestos, las pruebas correspondientes, y las incertidumbres que esto conlleva (Thomas & Cook, 2005).

Visualización: Stephan Diehl define visualización como el proceso de transformar información en una representación visual, con el objetivo de transmitir datos al cerebro a través del sistema visual humano mediante la elaboración de imágenes en la pantalla del computador, de tal forma que el usuario final, por ejemplo un científico o un ingeniero, sea capaz de descubrir datos de relevancia que se encuentran ocultos en dicha información (Diehl, 2007).

Según Thomas y Cook, la visualización de información amplía las capacidades cognitivas humanas en seis formas básicas:

1. Mediante el aumento de los recursos cognitivos.
2. Mediante la reducción de búsqueda.
3. Mediante la mejora del reconocimiento de patrones.
4. Mediante el apoyo a la fácil inferencia perceptual de relaciones que de otra manera son muy difíciles de inducir.
5. Mediante la monitorización perceptual de un gran número de eventos potenciales.
6. Proporcionando un medio manipulable que, a diferencia de los diagramas estáticos, permite la exploración de un espacio de valores paramétricos (Thomas & Cook, 2005).

Analítica visual: la analítica visual pretende unir técnicas de visualización de información con técnicas computacionales de transformación y análisis de datos, con el fin de crear sistemas de software que apoyen el proceso de razonamiento analítico (Thomas & Cook, 2005).

Para la creación de una visualización analítica, debe seguirse una serie de pasos ordenados, a saber:

1. Adquisición de datos: en esta etapa se recuperan datos de diferentes fuentes de información; por ejemplo, del código fuente, diseño, documentación de usuario, cambios de estado durante la ejecución, resultados de pruebas, entre otros.
2. Análisis: esta etapa consiste en aplicar técnicas de análisis a los datos obtenidos, de manera que se filtre la información relevante para centrarse en ella.
3. Visualización: finalmente, los datos se proyectan sobre un modelo visual, que luego es mostrado en la pantalla u otro medio como una imagen única o un conjunto de imágenes (Diehl, 2007).

Visualización de software: la visualización de software consiste en la visualización de artefactos relacionados con software y su proceso de desarrollo; es decir, el código fuente, requerimientos, documentación de diseño, cambios en el código fuente, informes de errores, entre otros. Por lo tanto, es el arte y la ciencia de la generación de representaciones visuales de diversos aspectos de software y su proceso de desarrollo (Diehl, 2007). La visualización de software es uno de los principales campos de aplicación de la visualización analítica en la ingeniería de software.

3.2 Analítica visual en la ingeniería de software

La analítica visual tiene varias utilidades en la ingeniería de software, entre ellas como soporte para evaluar la evolución del software en procesos de mantenimiento de sistemas. De acuerdo con Diehl (2007), algunos de los beneficios del uso de visualizaciones de software son los siguientes:

- Ahorro de tiempo y dinero
- Mejor comprensión del software
- Aumento de la productividad y la calidad

– Gestión de la complejidad
– Simplificación de la identificación de errores

A pesar de que la aplicación de la analítica visual en la ingeniería de software no es de uso común, ya existe todo un camino recorrido en este campo. A continuación se detallan algunos ejemplos de su aplicación, con el objetivo de dar una idea general del estado del arte:

– Un ejemplo en el que se utiliza la analítica visual como soporte para la ingeniería de software, es la visualización llamada *Revision Tree*, propuesta por Therón, González, García y Santos para ayudar en la gestión de la configuración del software en proyectos geográficamente distribuidos. En tal visualización se utiliza la información disponible en los repositorios de software acerca de las revisiones de los diferentes programadores, información que posteriormente es desplegada de manera que permite observar las contribuciones de los miembros del equipo en un mismo ítem de software a lo largo de varias revisiones, líneas base y periodos de tiempo prolongados (Therón, González, García, & Santos, 2007). Mediante el uso de esta visualización, estos autores indican que es posible obtener una gran cantidad de información de un vistazo, y no es necesaria una explicación detallada para descubrir datos de relevancia; de igual manera, es fácil hacer un seguimiento de las contribuciones al desarrollo de un elemento de software en particular y entender cómo ha evolucionado a lo largo del tiempo (Therón et al., 2007).
– Por otra parte, D'Ambros y Lanza proponen el uso de una técnica llamada arqueología de software, la cual consiste en utilizar visualizaciones para conocer la evolución del software. En este caso, los autores proponen utilizar esta técnica para entender, por un lado, la estructura general y la evolución de un sistema en términos de sus módulos o componentes de alto nivel; y, por otro, para entender el detalle de la estructura interna de los diferentes módulos, pasando de los directorios hasta el nivel de versiones de un archivo (D'Ambros & Lanza, 2009).
  En la visualización de la estructura general se pueden observar figuras rectangulares representando módulos de software (compuestos por directorios y archivos) de diferentes tamaños y tonalidades de colores, distribuidos a lo largo de una escala de tiempo, de forma que se puede obtener información como el tamaño de los módulos, los módulos con mayor cantidad de errores encontrados o bien los que cuentan con más revisiones; todas estas variables son indicadores de cuánto esfuerzo de desarrollo se asocia a cada uno de los módulos. Complementariamente, en la visualización de estructura detallada se considera la estructura interna de un módulo en particular, archivos e incluso errores; se utilizan rectángulos para mapear el número de revisiones, el número de líneas de código en la última versión del sistema y el número de errores presentes (D'Ambros & Lanza, 2009).
– Otro ejemplo del uso de visualizaciones de software para comprender la evolución de un sistema es el propuesto por German y Hindle, en el cual se utilizan de insumo datos históricos de CVS, correos electrónicos, reportes

de errores, bitácoras de cambios, entre otros, para posteriormente mostrar información relevante de los "rastros" del software por medio de visualizaciones sencillas, entre las que destacan un conjunto de gráficos de barras para observar relaciones como el número de archivos vs. el tiempo, y un visualizador que permite explorar la historia del proyecto mediante una colección de gráficos, entre estos uno para observar los cambios, otro con los autores y otro con información evolutiva de cuándo y quién realizó cambios en un archivo (German & Hindle, 2006).
- Otro uso que se le ha dado a las visualizaciones de software, es como método de soporte y ayuda para los desarrolladores en proyectos de gran tamaño, principalmente en la tarea de comprender la evolución de un componente de software en particular. Este es el caso del enfoque propuesto por Novais et al., en el cual se utiliza una herramienta llamada SourceMiner Evolution, que cuenta con varias visualizaciones interactivas para seleccionar y comparar dos versiones de una aplicación, mostrando las diferencias entre versiones a través de colores que resaltan la eliminación, agregación y características transferidas entre los diferentes elementos de software (Novais et al., 2012).

Muchas de las técnicas y elementos visuales que se mencionan en los ejemplos anteriores de visualizaciones de software forman parte de las características funcionales de GridMaster, visualización que se utilizará como caso de aplicación en las secciones siguientes de este documento.

4 Analítica visual en la Web

Las herramientas de analítica visual tradicionalmente han sido implementadas en ambientes de escritorio; sin embargo, con el auge de las tecnologías web y todas las facilidades tecnológicas disponibles, se ha venido incursionando en el desarrollo de herramientas web que emplean técnicas de visualización analítica (D'Ambros, Lanza, Lungu, & Robbes, 2011). Partiendo de este hecho, D'Ambros et al. enumeran una serie de promesas y retos relacionados con la migración de visualizaciones de software a entornos web.

Algunas de las promesas más llamativas son las siguientes:

- Disponibilidad: migrar las herramientas de visualización de software a la nube las pone a disposición de más personas, en comparación con las versiones de escritorio.
- Colaboración: al migrar a la web, se facilita el proceso de colaboración.
- Despliegue selectivo y retroalimentación: se pueden desplegar selectivamente cambios a un grupo de usuarios y medir su efecto (D'Ambros et al., 2011).

Por otro lado, la implementación de una visualización en un entorno web debe enfrentar los siguientes retos:

- Privacidad: la información confidencial sobre los sistemas de software no debe estar disponible para las personas no autorizadas.

- Rendimiento y escalabilidad: las visualizaciones colaborativas web deben servir grandes cantidades de datos a muchos usuarios a la vez, lo cual puede convertirse en un cuello de botella que afecte a todos los usuarios.
- Compatibilidad de navegadores: las aplicaciones web tienen que hacerle frente a los problemas de compatibilidad entre exploradores (D'Ambros et al., 2011).

5 GridMaster: una herramienta para la visualización de la evolución del software

GridMaster es parte de un conjunto de visualizaciones para revelar hallazgos de la evolución del software; fue diseñada por el Msc. Antonio González Torres, especialista en el campo de la analítica visual de software y análisis de datos. El conjunto de visualizaciones tienen el objetivo de facilitar el proceso de mantenimiento de proyectos de software. La arquitectura que proponen González et al. consiste en cinco grandes componentes:

1. El motor de extracción de conocimiento basado en evolución
2. El software de hechos de evolución
3. El motor de transformación de datos para los modelos de visualización
4. Las visualizaciones de la evolución del software
5. El usuario en sí mismo (González et al., 2011)

A grandes rasgos, esta arquitectura funciona de la siguiente manera: el motor de extracción de conocimiento extrae los datos de detalles evolutivos de los diferentes repositorios del proyecto; luego, a estos datos se les aplica un software exhaustivo de análisis evolutivo, cuyos resultados se almacenan en la base de datos de hechos de evolución del software. Los datos que se almacenan corresponden a las líneas de tiempo de los componentes de software, las métricas de evolución, las relaciones sociotécnicas, y algunas relaciones arquitecturales/estructurales, como por ejemplo herencia, implementación de interfaces y correlaciones entre datos estructurales y métricas. Posteriormente, el motor de transformación de datos transforma estos datos de hechos en las estructuras de datos adecuadas para que sean utilizadas por el software de visualización de la evolución.

Las visualizaciones incluidas en la propuesta de González et al. forman parte de un plugin de Eclipse para proyectos de software que se desarrollan en un ambiente de escritorio, y utilizan una representación matricial, una línea de tiempo y vistas sociotécnicas que son apoyadas por un gráfico de red social y el uso de diferentes colores para representar las contribuciones de los programadores.

Ahora bien, con respecto a las visualizaciones, es importante mencionar que la vista principal llamada GridMaster corresponde a una representación matricial donde las filas de la matriz se asocian a paquetes y componentes de software, mientras que las columnas son unidades de tiempo. Cada una de las celdas formadas se utiliza para representar correlaciones entre los diferentes paquetes o componentes de software; por ejemplo, las contribuciones de los programadores,

la creación de componentes de software, la adición o remoción de herencia o relaciones de implementación de interfaces, y métricas de los diferentes componentes de software.

Entre las principales interacciones que esta vista matricial permite realizar, se encuentran el ampliar y reducir la imagen, el reacomodo de los elementos de la estructura del proyecto, la capacidad de filtrar nodos de la estructura, seleccionar un año en específico de la línea de tiempo para observar datos de los meses de ese año, y además permite seleccionar cómo se desplegarán las métricas y los programadores al seleccionar entre dos representaciones: una relativa que considera el componente de software con el mayor valor de la métrica para la altura de la gráfica, y una absoluta que considera el valor de la mayor métrica relacionada con un componente de software.

González et al. proponen dos casos de aplicación concretos, relacionados con el mantenimiento de software, que hacen uso de GridMaster:

1. Previo a la corrección de un error, mediante la vista matricial, se escoge un periodo de tiempo conveniente con respecto al trabajo realizado en el elemento de software relacionado con el error, y luego se puede inferir cuál es el programador que mejor se adapte a la necesidad del momento para llevar a cabo el arreglo requerido.
2. Conocer los detalles de implementación de cara a una refactorización de un componente de software. En este caso, basta con seleccionar el componente de software y expandirlo para examinar los detalles de sus relaciones de herencia e implementación de interfaces (González et al., 2011).

6 Propuesta de implementación de GridMaster para IDE de nube

El alcance de este trabajo está limitado a crear un prototipo funcional de la visualización GridMaster, que pueda ser utilizado desde la web; con base en esto, se establece como restricción principal el uso exclusivo de tecnologías web para hacer el despliegue gráfico de la visualización y controlar la interacción con el usuario. Quedan por fuera las etapas de adquisición y análisis de datos, ya que los datos procesados fueron obtenidos de la versión de GridMaster hecha para IDE de escritorio.

Además, se establece como premisa que se pretende en uno o varios proyectos de investigación posteriores seguir el hilo de este trabajo, en los cuales se requeriría la identificación de IDE de nube candidatos para realizar la adaptación de esta propuesta como un plugin en el ambiente de desarrollo seleccionado. Con base en dicho supuesto, y al no tener identificado el entorno de desarrollo, surge como requerimiento que la implementación debe ser independiente de plataforma.

Para la implementación se desea utilizar D3 (Data-Driven Documents), una biblioteca de Javascript para hacer la representación gráfica de la visualización. D3 construirá la visualización esencialmente a través de SVG (Scalable Vector

Graphics), una tecnología soportada por los navegadores de uso más común. La base de datos empleada será MySQL, ya que además de ser una base de datos multiplataforma y de código libre, para este motor ya se cuenta con una base de datos poblada con datos de varios proyectos que servirán como casos de prueba. Para acceder a la base de datos se empleará PHP, y para transferir los datos requeridos por la visualización se utilizará la notación JSON (JavaScript Object Notation).

Durante las primeras pruebas realizadas con la biblioteca D3, se evidencia que los principales inconvenientes de su uso están relacionados con el tiempo de respuesta en el despliegue gráfico de la visualización, especialmente al darse la interacción con el usuario. Debido a lo anterior, el desarrollo de la visualización se realiza dando énfasis en la reducción de los tiempos de respuesta del despliegue de la visualización y la interacción con el usuario, lo cual se logró principalmente reduciendo la cantidad de elementos gráficos que se dibujan sobre el lienzo de trabajo, así como proveyendo la menor cantidad de datos a la visualización y tan preprocesados como fuera posible.

Otro factor que durante el proceso de desarrollo se identificó como una variable que afectaba el tiempo de respuesta, era el realizar ciertos cálculos en las consultas que se enviaban a la base de datos MySQL, ya que en muchas ocasiones estos eran costosos y repetitivos. Para solventar esta situación, parte del procesamiento intermedio requerido para algunas consultas complejas se realiza en código PHP que es ejecutado en el servidor web.

7 Análisis de los resultados de la implementación de un prototipo funcional

7.1 Prototipo funcional.

Luego de algunas semanas de desarrollo, se logra contar con una versión funcional del prototipo de la visualización GridMaster en versión web. A continuación se muestran algunas imágenes donde se resaltan los puntos más relevantes de dicho prototipo.

En la Figura 1 se muestra el despliegue de las contribuciones en modo relativo para los paquetes principales de un proyecto; también se muestra el selector de proyectos, así como el resto de las opciones del menú. El modo relativo normaliza las contribuciones tomando como referencia cantidad de revisiones de cada paquete o archivo en su respectiva unidad de tiempo.

En la Figura 2 se muestra el despliegue de las contribuciones en modo absoluto para los paquetes principales de un proyecto. El modo absoluto normaliza las contribuciones tomando como referencia la unidad de tiempo con más revisiones en todo el proyecto.

En la Figura 3 se muestra el detalle de un año expandido, para señalar la actividad de contribuciones de los programadores en cada uno de los meses de ese año.

En la Figura 4 se muestra el detalle de los componentes de software, en el cual se subdivide el espacio de la celda para representar cada mes. Las contribuciones

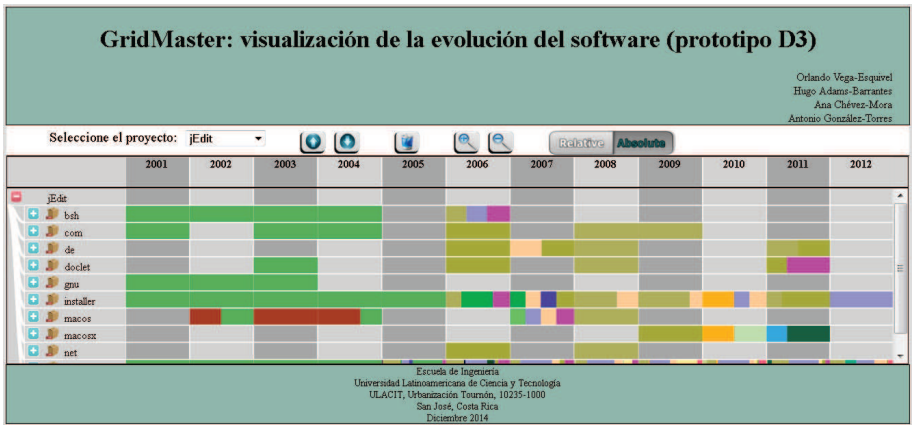

Fig. 1. Paquetes principales, contribuciones en modo relativo y opciones del menú. Fuente: Elaboración propia.

se normalizan en relación con el máximo número de métodos que haya tenido cada componente de software.

En la Figura 5 se muestra la representación del inicio (óvalo verde) y finalización (óvalo rojo) de las relaciones de herencia de clases e implementación de interfaces.

7.2 Utilización de la herramienta.

Mediante la utilización del prototipo funcional, se nota cómo es posible obtener indicios de eventos ocurridos durante el ciclo de vida de los proyectos que se visualizan con la herramienta. Por un lado, se obtienen indicios de ciertos patrones comunes en los proyectos utilizados como caso de ejemplo patrones que se asume podrían darse debido a las características inherentes de las comunidades de desarrollo de código libre:

1. Al inicio del desarrollo todas las contribuciones son realizadas por uno o pocos programadores.
2. A medida que avanza el proyecto en el tiempo se integra una mayor cantidad de programadores.
3. Los programadores que interactuaban más dentro del proyecto, poco a poco van dejando de contribuir en él, y en muchos casos parecen abandonarlo del todo.
4. También se vislumbra la variedad de participación entre los programadores: algunos suben gran cantidad de revisiones y de forma más frecuente, mientras que otros suben pocas revisiones y en forma esporádica.

Por otro lado, también se observan eventos que deben ser interpretados de acuerdo con el contexto de la lógica de representación de la herramienta, como por ejemplo:

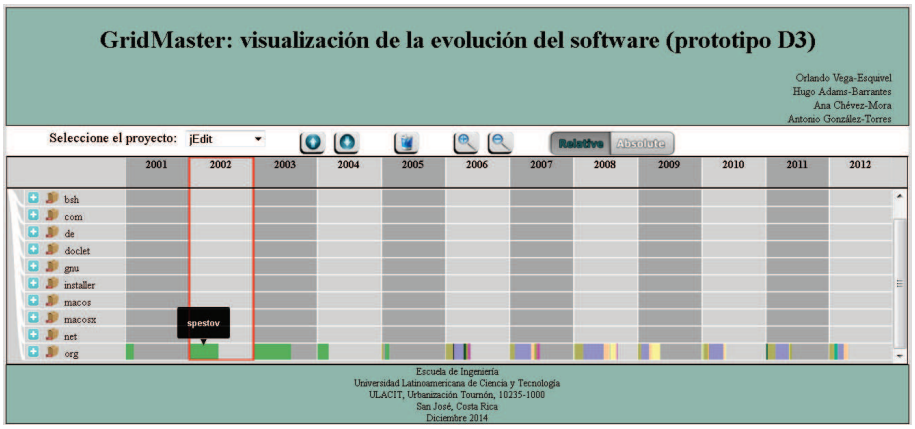

Fig. 2. Paquetes principales, contribuciones en modo absoluto y opciones del menú. Fuente: Elaboración propia.

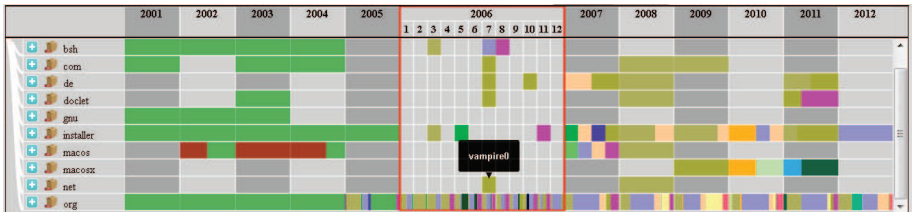

Fig. 3. Año expandido para detallar la actividad por mes. Fuente: Elaboración propia.

1. Cuando cesa la actividad en un elemento del árbol de la visualización, no se puede asumir que hayan llegado a un estado estable y sin cambios, sino que este podría haber sido borrado o trasladado a otro punto en la jerarquía. Por ello, es conveniente realizar este tipo de análisis utilizando la visualización en conjunto con la estructura actual del proyecto.
2. Cuando para una clase se finaliza una relación de herencia o implementación, es de esperar que se inicie una nueva relación que reemplaza la anterior.

8 Conclusiones

Gracias a los avances tecnológicos como la computación en la nube, ahora es más común encontrar equipos de desarrolladores distribuidos en diferentes locaciones geográficas, utilizando ambientes de desarrollo colaborativo en la nube para trabajar de manera conjunta y construir soluciones de software. Esta situación, sin embargo, genera una mayor necesidad, tanto para el director de proyectos como para los diferentes desarrolladores, de contar con herramientas que les permita conocer el estado del sistema de software, y en particular conocer detalles sobre su evolución, para ayudarse en las tareas de mantenimiento del software.

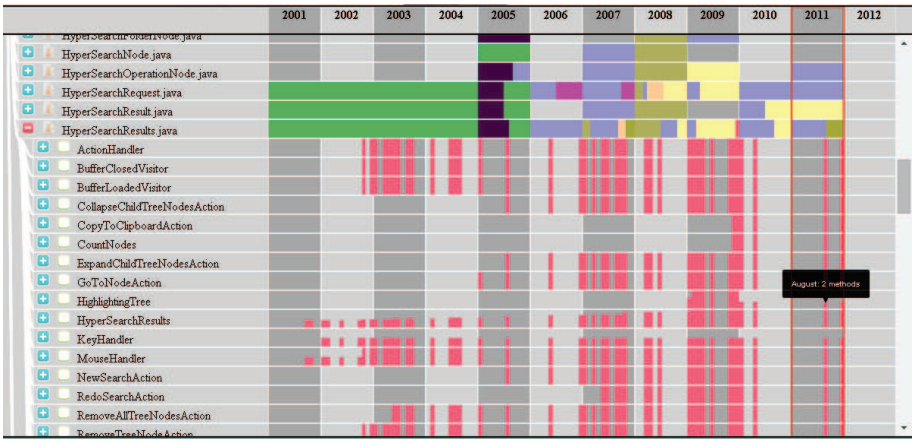

Fig. 4. Contribuciones de archivos e ítems de software. Fuente: Elaboración propia.

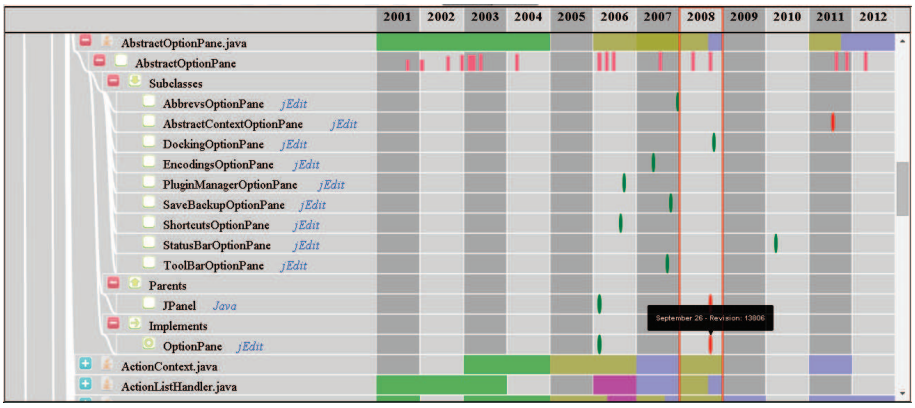

Fig. 5. Relaciones de herencia e implementación. Fuente: Elaboración propia.

La analítica visual une técnicas de visualización de información con técnicas de transformación y análisis de datos, con el fin de crear sistemas de software que apoyen el proceso de razonamiento analítico (Thomas & Cook, 2005). Estas herramientas de analítica visual tradicionalmente han sido implementadas en ambientes de escritorio; sin embargo, con el auge de la era web, es necesario contar con todas estas facilidades en los nuevos entornos de desarrollo web que se han originado.

Una visualización como GridMaster traería un gran valor agregado a la gestión de los proyectos de desarrollo de software, al facilitar la asignación del trabajo que debe realizarse y mejorar la efectividad del tiempo dedicado por cada programador. Dicho potencial puede ser explotado principalmente en entornos altamente colaborativos; tal es el caso de las comunidades de desarrollo de proyectos de código libre, donde puede haber gran cantidad de participantes

con diversa dedicación de tiempo, frecuencia de participación y especialización técnica.

Derivado del hecho de que la visualización analítca por definición depende de un robusto e interactivo despliegue gráfico, su utilización en entornos web presenta claras limitaciones que deben considerarse a la hora de realizar una implementación de este tipo. Sin embargo, una vez concluido el prototipo funcional relacionado con este trabajo se afirma que las tecnologías web seleccionadas (D3, Javascript y PHP) tienen el potencial requerido para hacer el despliegue de una visualización analítica de complejidad similar a la seleccionada como caso de ejemplo (GridMaster).

9 Recomendaciones para futuras investigaciones

A pesar de la exitosa implementación del prototipo funcional realizado como parte de esta investigación, aún es posible mejorar su implementación; sin embargo, se considera que está listo para ser empleado en la siguiente etapa de investigación, la cual corresponde a utilizar el prototipo presentado para incorporarlo en un entorno integrado de desarrollo en la nube.

En una eventual implementación donde se busque mejorar el tiempo de respuesta de la visualización, debe trabajarse la representación gráfica en una forma más manual para obtener mayor control de los elementos visuales, de tal manera que sea posible desplegar únicamente los elementos gráficos que pueden ser apreciados por el usuario en un momento dado. Para el caso del prototipo desarrollado, automáticamente se invoca la aplicación de modificaciones en el despliegue gráfico de todos los elementos expandidos en el árbol de la visualización, aun cuando dichos elementos exceden el área de visualización del usuario.

También, para una eventual mejora en el tiempo de respuesta de la visualización, se recomienda realizar ciertas variantes en el modelo de base de datos empleado, para que este favorezca el rendimiento de las consultas requeridas por la visualización, ya que el modelo utilizado en el presente trabajo originalmente fue concebido para realizar un procesamiento por lotes previo a la utilización de la visualización, lo cual no es factible en entornos web. Aunque por motivos de tiempo no fue posible adaptar el modelo de base de datos, migrar los datos existentes y emplear el modelo propuesto con este prototipo, en la Figura 6 se muestra la propuesta de un modelo de base de datos que podría disminuir el tiempo de respuesta de las consultas realizadas por el prototipo web.

Referencias

Albaroodi, h., Hala A.1, Manickam, s., Selvakumar1, & Aboalmaaly, e., Mohammed Faiz1. (2013). The classification and arts of open source cloud computing: A review. *Advances in Information Sciences & Service Sciences*, *5*(16), 16 - 25. Retrieved from `http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=97946063&lang=es&site=ehost-live` pages 2, 3

D'Ambros, m., Marco1, & Lanza, M. (2009). Visual software evolution reconstruction. *Journal of Software Maintenance and Evolution: Research and Practice*, *21*(3), 217 - 232. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=40631755&lang=es&site=ehost-live pages 1, 5

D'Ambros, m., Marco1, Lanza, m., Michele1, Lungu, l., Mircea2, & Robbes, r., Romain3. (2011). On porting software visualization tools to the web. *International Journal on Software Tools for Technology Transfer*, *13*(2), 181 - 200. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=59460295&lang=es&site=ehost-live pages 6, 7

Diehl, S. (2007). Software visualization - visualizing the structure, behaviour, and evolution of software. *Berlin Heidelberg: Springer-Verlag*, 1-15. pages 3, 4

Ferzund, j., Javed1, Yasrab, r., Robail1, & Razzaq, s., Saad2. (2014). Web 2.0 and collaborative software development. *International Journal of Software Engineering and Its Applications*, *8*(7), 107 - 120. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=97380407&lang=es&site=ehost-live pages 3

German, d., Daniel M.1, & Hindle, a., Abram1. (2006). Visualizing the evolution of software using softchange. *International Journal of Software Engineering & Knowledge Engineering*, *16*(1), 5 - 21. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=20060847&lang=es&site=ehost-live pages 6

González, A., Therón, R., García, F., Wermelinger, M., & Yu, Y. (2011). Maleku: an evolutionary visual software analytics tool for providing insights into software evolution. *Valladolid: 27th International Conference on Software Maintenance*, 25-30. pages 2, 7, 8

Karanam, b., Madhavi1, & Akepogu, a., Anand Rao2. (2011). Framework for visualizing model-driven software evolution – its evaluation. *International Journal of Software Engineering and Its Applications*, *5*(2), 135 - 148. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=61135668&lang=es&site=ehost-live pages 1

Novais, . r., Renato1, Nunes, c.-r., Camila3, Lima, c., Caio1, Cirilo, e.-r., Elder3, Dantas, f.-r., Francisco3, Garcia, a.-r., Alessandro3, & Mendonça, m., Manoel1. (2012). On the proactive and interactive visualization for feature evolution comprehension: An industrial investigation. *ICSE: International Conference on Software Engineering*, 1044 - 1053. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=78198172&lang=es&site=ehost-live pages 6

Therón, R., González, A., García, F. J., & Santos, P. (2007). The use of information visualization to support software configuration management. *Valladolid: IFIP International Federation for Information Processing*, 317-

331. pages 5

Thomas, J. J., & Cook, K. A. (2005). Illuminating the path. the research and development agenda for visual analytics. *National Visualization and Analytics Center*. pages 2, 3, 4, 12

Fig. 6. Modelo físico de base de datos propuesto como posible mejora. Fuente: Elaboración propia.