Técnica para la visualización de la evolución de cambios en el código fuente

Joan S. Lopez Riaño
Facultad de Ingeniería
Universidad Nacional de Colombia
Email: jslopezr@unal.edu.co

Felipe Restrepo Calle PhD
Facultad de Ingeniería
Universidad Nacional de Colombia
Email:ferestrepoca@unal.edu.co

John A. Guerra Gómez PhD
Facultad de Ingeniería
Universidad de los Andes
Colombia
Email:john.guerra@gmail.com

***Resumen*—En el campo de la visualización de la evolución del software, los trabajos que apuntan a facilitar la visualización de dicho proceso no han sido suficientes, así como las implementaciones concretas. El presente proyecto pretende desarrollar e implementar una técnica para la visualización de la evolución de los cambios en el código fuente usando principalmente el análisis de repositorios de software y las métricas relacionadas con los mismos.**

***Palabras claves*—Análisis visual, Evolución de Software, Métricas de Código Fuente, Técnicas de Visualización, Visualización de Software.**

I. Introducción

En ingeniería de Software se entiende el concepto metáfora de visualización como "aquella analogía que subyace a una representación gráfica de una entidad o concepto abstracto con el objetivo de transferir propiedades del dominio específico de la representación gráfica al dominio de la entidad abstracta o concepto que se quiere comprender"[1]. Existen diversos ejemplos de metáforas de visualización aplicadas al código fuente, de las cuales algunas ya se han establecido como clásicos dentro del campo de estudio, como lo son las metáforas de ciudades[2], hasta otras propuestas más experimentales o artísticas como las de representaciones con plumas de aves[3] o sistemas solares[4]. En trabajos recientes se puede evidenciar que no son muchas las propuestas de visualizaciones enfocadas a representar características relacionadas con la evolución del software y en igual medida también son escasas las implementaciones concretas de dichas propuestas[5]. Al contemplarse que se llevará a cabo una implementación de la técnica de visualización, es necesario tener en cuenta aspectos de la experiencia de usuario al momento de navegar por los conjuntos de información que proveerá la visualización resultante, pues para mejorar la asimilación y comprensión de los conceptos representados por medio de metáforas el diseño de componentes interactivos es primordial[6].

II. Estado del arte

A. *Software Evolution o la naturaleza evolutiva del Software*

Desde los inicios de la computación moderna y la producción del software los procesos mediante los cuales se creaban programas informáticos fueron importados desde áreas del conocimiento como la Ingeniería Civil. Iniciando con modelo de desarrollo en cascada propuesto en 1970 y mas adelante pasando por los modelos cíclicos iterativos se llegó a acuñar el termino Software Evolution en los años noventa del siglo XX. Finalmente, en el año 2001 fue gracias a la llegada de las metodologías ágiles de desarrollo[7] que se analizó con rigurosidad académica el fenómeno de la naturaleza evolutiva del software, diferenciando esta característica de las recurrentes tareas de mantenibilidad al final del proceso de desarrollo. En términos prácticos la expresión Software Evolution sirve para referirse al conjunto de tareas más comunes en la producción de software relacionadas con el mantenimiento en todas las etapas de la producción, esto incluye la corrección de errores, inclusión de mejoras y nuevas funcionalidades, optimización y adaptabilidad a nuevas tecnologías.

Dado que las labores de mantenimiento siempre tenían que ser consideradas en cualquier proyecto se llegó a la conclusión de que el software en esencia es de naturaleza evolutiva y no estática como se creía en los inicios de la Ingeniería de Software.

Las implicaciones de esta naturaleza evolutiva justificarían la necesidad de proveer visualizaciones de código que abarquen mas de un instante de tiempo pues los cambios que tuviesen los proyectos también deberían poder ser visualizados en una escala temporal[8]. En la actualidad la evolución de un proyecto de software, en especial de código abierto puede ser analizada gracias a sistemas de versionamiento como Subversion, Git, Mercurial, CVS, entre otras.

B. *Métricas del código fuente*

Gracias al aporte de [9] se podrían generalizar la siguientes métricas del código fuente aplicadas a proyectos de la categoría FLOSS:

Métricas de tamaño		
Abreviatura	**Descripción (en Inglés)**	**Descripción (Castellano)**
LOC	*Lines Of Code*	Lineas de código
AMLOC	*Average Method Lines Of Code*	Promedio de lineas de código por método
-	*Total Number of Modules or Classes*	Cantidad total de módulos o clases
Métricas de mantenibilidad, flexibilidad y entendimiento de código		
Abreviatura	**Descripción (en Inglés)**	**Descripción (Castellano)**
NOA	*Number of Attributes*	Número de atributos por clase
NOM	*Number of Methods*	Número de métodos por clase
NPA	*Number of Public Attributes*	Número de atributos públicos
NPM	*Number of Public Methods*	Número de métodos públicos
ANPM	*Average Number of Parameters per Method*	Número promedio de parámetros por método
DIT	*Depth of Inheritance Tree*	Profundidad del árbol de herencia entre clases
NOC	*Number of Children*	Número de hijos por clase
RFC	*Response for a Class*	Respuestas para una clase
ACCM	*Average Cyclomatic Complexity per Method*	Promedio de complexidad ciclomática por método

C. Visualización de Software

Comprende el área de la ingeniería de software que se encarga de desarrollar metodologías y herramientas visuales para entender mejor el diseño y funcionalidad de una aplicación usando como base el análisis a profundidad de artefactos como el código fuente (principalmente), la información extraída de repositorios de software, entre otros.

Sin embargo una apropiada definición se extendería al relacionar otros aspectos que vayan más allá de la visualización de artefactos de software y el proceso de desarrollo del mismo. En el área de investigación de la visualización de software se encaminan esfuerzos para el desarrollo de propuestas que permitan la visualización de estructura, comportamiento y la evolución de todas las piezas que comprenden un programa[1].

El proceso según el cual se construyen las visualizaciones podría resumirse como : Adquisición de información, luego análisis de la misma y finalmente el mapeado de lo analizado en una representación visual, la cual puede estar compuesta por una o mas imágenes, incluso hasta animaciones[1].

Aplicaciones de la visualización de software

Según investigaciones y encuestas enfocadas en el estudio de los principales beneficios de la implementación de metodologías de visualización en proyectos de software, se encuentran[10]:

- Reducción en los costos de producción del software
- Mejor comprensión del sistema
- Incremento en la productividad
- Gestión de la complejidad
- Asistencia en la búsqueda de errores
- Mejoramiento de la calidad

D. Técnicas de visualización según el enfoque

En [11] se evidencia un trabajo por llevar a cabo una revisión sistemática de la literatura que permitiera dar cuenta de una clasificación actual para los trabajos de investigación en herramientas aplicadas a la visualización de la evolución del software.

La propuesta establece la siguiente clasificación de los tipos de técnicas de visualización implementadas y los porcentajes de la distribución de la cantidad de los mismos:

- Basada en grafos - 45%
- Basada en notaciones - 3%
- Basada en matrices - 29%
- Basada en metáforas - 7%
- Otras - 16%

Para una clasificación que dividiera los trabajos encontrados respecto a su objetivo o concepto a representar los autores proponen las siguientes categorías:

- Visualización enfocada en los cambios sobre artefactos
- Visualización enfocada en los cambios sobre métricas
- Visualización enfocada en los cambios sobre características
- Visualización enfocada en los cambios sobre la arquitectura

Herramientas que soportan la visualización de software

En [5] se llevó a cabo una revisión sistemática de la literatura donde a partir del análisis de 52 investigaciones se pudieron identificar 28 técnicas de visualización así como 33 herramientas o implementaciones de dichas técnicas. Los resultados del estudio, fueron clasificados según la técnica, el enfoque (estructura, jerarquía, artefactos, dependencia, evolución y métricas) , el tipo de análisis (estático o dinámico), el formato (grafos abstractos o metáforas) entre otras características y si existía o no una herramienta de software concreta que implementara dicha técnica o metodología de visualización.

E. Visualizaciones aplicadas a la evolución del Software

Dentro de los trabajos recientes mas relevantes relacionados con el tema de evolución del software, se encuentra *Evowave* [12]. Dicho proyecto resalta por la versatilidad para representar distintos conjuntos de información relacionados con el proceso de desarrollo y evolución de un proyecto de software, involucrando distintas métricas que van desde el análisis de cambios de artefactos hasta los autores que se han involucrado en el desarrollo. La metáfora de las ondas es análoga al fenómeno de la dispersión de las mismas a lo largo del tiempo de forma circular.

F. La interactividad en las técnicas de visualizaciones implementadas

Dentro de el conjunto de herramientas que implementan técnicas de visualización, solo algunas de ellas tienen un componente interactivo, es decir, que el usuario final puede manipular la visualización resultante, bien sea modificando su conjunto de origen de datos, los enfoques de proximidad o simplemente navegando por la visualización. Dentro de los proyectos que mas resaltan por su interactividad encontramos:

[13] –Una herramienta para la visualización del grafo de llamadas entre métodos. Desarrollado en la plataforma Processing la cual permite implementar una interfaz de usuario interactiva.
[14] – CuboidMatrix, implementa una visualización dentro de un cubo, donde uno de sus ejes representa el tiempo, permitiendo de esta manera navegar por un conjunto de datos desde la interfaz de usuario.
[15] – Metavis , implementa la metáfora de nubes, o tags para navegar por conceptos relevantes marcados en un conjunto de datos.
[16] – CodeSurveyor, su interactividad está dada por la navegación que se puede tener a lo largo de la visualización, la cual implementa la metáfora de un mapa cartográfico permitiendo realizar acercamientos a distintos niveles de detalle.
[17] – A través de una visualización que mezcla la metáfora de la ciudad con la de mapas de calor, este trabajo implementa elementos de navegación a lo largo de un entorno 3D.
[18] –Al igual que el trabajo con mapas de calor, esta propuesta implementa la metáfora del bosque y árboles donde el usuario puede navegar por el escenario 3D generado por el proceso de visualización.

G. *Conclusiones del estado del arte*

1. Es notable que la mayoría de propuestas se decantan por el uso de representaciones gráficas abstractas (nodos, grafos, diagramas) por sobre la implementación de metáforas.
2. Son escasas las propuestas que implementan visualizaciones enfocadas a tareas de la evolución del software con componentes interactivos enfocados en el usuario final.
3. No son muchas las propuestas dentro de la categoría de estudio de la evolución del software que implementen metáforas figurativas. Si bien existen muchas metodologías y trabajos que se enmarcan dentro del área de visualización de software, no todas las propuestas implementan herramientas concretas que permitan poner en práctica las mismas.

III. Metodología

Para la efectiva implementación del proyecto se optará por una metodología explotaría, es decir, que se partirá de una comprensión de la actualidad en el tema de visualización de Software , la implementación de representaciones visuales y como se ha tratado el aspecto de la naturaleza evolutiva del código fuente.

Con el propósito de llevar a cabo lo definido en el objetivo principal (diseñar una técnica para la visualización de la evolución del código fuente) se definirán un conjunto de posibles propuestas de representaciones visuales las cuales serán validadas mediante la construcción de una aplicación de software por cada una. Finalmente se espera aplicar la metodología propuesta a algunos casos de estudio y a partir de los productos obtenidos poder elaborar y exponer las conclusiones del trabajo realizado.

Los cuatro objetivos específicos se repartirán en cuatro fases de evolución del proyecto, las cuales se pueden resumir como: Levantamiento del estado del arte en el tema de visualizaciones enfocadas en la evolución del software, definición de la metáfora visual y sus características, implementación de la metodología y finalmente validación de la misma mediante algunos casos de estudio

IV. Fases del proyecto

Fase I: Construcción del estado del arte
Representa el esfuerzo inicial por comprender el estado del arte en el tema de visualizaciones de software, la evolución del mismo y la implementación de representaciones visuales para la representación de las distintas características a analizar, clasificándolas según su relevancia y/o utilidad.

Fase II: Diseño de la técnica de visualización de la evolución de código fuente
Contempla la exploración de varias propuestas para la representación visual de la técnica a diseñar, escogiendo aquellas que cuenten con las bases más sólidas que justifiquen su usabilidad y pertinencia. De igual manera se definirá el conjunto de características a extraer tanto de los artefactos de código fuente como de los repositorios de versionamiento de cada proyecto.

Fase III: Implementación de la técnica
Definida la representación visual y las características del código fuente que se desea representar, se procederá a la implementación de dicha técnica mediante la construcción de una aplicación que tenga como objetivo generar las visualizaciones sobre los artefactos de código de un proyecto open source como caso de estudio. Para la realización del prototipo se utilizará la metodología de desarrollo SCRUM1 simplificada, la cual, aplicada a un equipo unipersonal con sprints o iteraciones semanales permitirá agilizar la generación cada semana de nuevas versiones que se acerquen más a la visualización deseado.

Fase IV: Validación de la técnica
En esta fase final se planea realizar un análisis exploratorio que tenga como objetivo sustentar la justificación de la metáfora visual seleccionada al estudiar distintas visualizaciones generadas por medio de la técnica propuesta aplicada a algunos artefactos de código fuentes que serán tratados como casos de estudio.

V. Conclusiones

Se puede predecir que unos de los aspectos mas relevantes y que mas trabajo requiere la propuesta, es el de la validación respecto a la utilidad y usabilidad de la técnica propuesta. Por esta razón urge la realización de iteraciones de la implementación que permitan una retroalimentación constante de la propuesta visual a desarrollar.

Finalmente, mas allá de aportar una visualización más al campo de la ingeniería de software, se busca realizar un aporte que permita relacionar las áreas del análisis visual y la evolución del software con perspectivas de facilitar las tareas de la comprensión del proceso de desarrollo de un proyecto de software.

Referencias

[1] S. Diehl, Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software. Springer Science & Business Media, 2007.

[2] R. Wettel y M. Lanza, «Visualizing Software Systems as Cities», en 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007, pp. 92-99.

[3] F. Beck, «Software Feathers figurative visualization of software metrics», en 2014 International Conference on Information Visualization Theory and Applications (IVAPP), 2014, pp. 5-16.

[4] H. Graham, H. Y. Yang, y R. Berrigan, «A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics», en Proceedings of the 2004 Australasian Symposium on Information Visualisation - Volume 35, Darlinghurst, Australia, Australia, 2004, pp. 53–59.

[5] A. Cruz, C. Bastos, P. Afonso, y H. Costa, «Software visualization tools and techniques: A systematic review of the literature», en 2016 35th International Conference of the Chilean Computer Science Society (SCCC), 2016, pp. 1-12.

[6] J. L. Cybulski, S. Keller, y D. Saundage, «Interactive Exploration of Data with Visual Metaphors», Int. J. Softw. Eng. Knowl. Eng., vol. 25, n.o 2, pp. 231-252, mar. 2015.

[7] «Manifiesto por el Desarrollo Ágil de Software». [En línea]. Disponible en: http://agilemanifesto.org/iso/es/manifesto.html. [Accedido: 13-nov-2016].

[8] T. Mens y S. Demeyer, Software Evolution. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[9] Paulo Roberto Miranda Meirelles, «Monitoramento de métricas de código-fonte em projetos de software livre», 2013.

[10] Stefan-Lucian Voinea, Software evolution visualization. Eindhoven: Technische Universiteit Eindhoven, 2007.

[11] H. B. Salameh, A. Ahmad, y A. Aljammal, «Software evolution visualization techniques and methods - a systematic review», en 2016 7th International Conference on Computer Science and Information Technology (CSIT), 2016, pp. 1-6.

[12] R. C. Magnavita, «Evowave: A Multiple Domain Metaphor for Software Evolution Visualization», jun. 2016.

[13] M. D. Shah y S. Z. Guyer, «An Interactive Microarray Call-Graph Visualization», en 2016 IEEE Working Conference on Software Visualization (VISSOFT), 2016, pp. 86-90.

[14] T. Schneider, Y. Tymchuk, R. Salgado, y A. Bergel, «CuboidMatrix: Exploring Dynamic Structural Connections in Software Components Using Space-Time Cube», en 2016 IEEE Working Conference on Software Visualization (VISSOFT), 2016, pp. 116-125.

[15] L. Merino, M. Ghafari, O. Nierstrasz, A. Bergel, y J. Kubelka, «MetaVis: Exploring Actionable Visualization», en 2016 IEEE Working Conference on Software Visualization (VISSOFT), 2016, pp. 151-155.

[16] N. Hawes, S. Marshall, y C. Anslow, «CodeSurveyor: Mapping large-scale software to aid in code comprehension», en 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT), 2015, pp. 96-105.

[17] O. Benomar, H. Sahraoui, y P. Poulin, «Visualizing software dynamicities with heat maps», en 2013 First IEEE Working Conference on Software Visualization (VISSOFT), 2013, pp. 1-10.

[18] U. Erra, G. Scanniello, y N. Capece, «Visualizing the Evolution of Software Systems Using the Forest Metaphor», en 2012 16th International Conference on Information Visualisation, 2012, pp. 87-92.