Conciencia Tecnológica No. 43, Enero-Junio 2012

Prototipo para Almacenar y Recuperar Métricas de Software

Investigación

Dr. Enrique Luna Ramírez[1], M.A.T.I. Humberto Ambriz D.[1], M.T.I. J. Antonio Nungaray O.[1]
M.C. David Hernández Chessani[2], L.I. Miguel Ángel Rodríguez Reyes[2]
[1]Instituto Tecnológico El Llano Aguascalientes, México, Carr. Ags.-S.L.P. Km. 18, El Llano, Ags.
elunaram@hotmail.com, humbertoambriz@hotmail.com, aenr25@hotmail.com
[2]Universidad Tecnológica de Aguascalientes, México, Blvd. Juan Pablo II No. 1302, Ags., Ags.
dhernandez@utags.edu.mx, mrodrey@ utags.edu.mx

Resumen

En este artículo se presenta un prototipo de repositorio para almacenar y recuperar métricas de software, extraídas de proyectos de desarrollo de software mediante herramientas de minería de datos. El repositorio fue diseñado para almacenar las métricas en forma de metadatos y poder recuperarlas mediante alias de identificadores. Para mostrar la operabilidad del repositorio, se presentan ejemplos relativos al proceso de desarrollo de software, en particular, a la construcción de modelos de estimación del tamaño de un proyecto de desarrollo de software.

Palabras clave: métricas de software, minería de datos.

Abstract

In this paper, it is presented a repository prototype to store and retrieve software metrics, extracted from software development projects through data mining tools. The repository was design to store metrics as metadata and be able to retrieve them through alias of identifiers. To show the repository operability, examples relative to the software development process are presented, particularly, relative to model construction for estimating the size of a software development project.

Key words: software metrics, data mining.

Introducción

Desde hace mucho tiempo se ha discutido el tema de la medición en el proceso de desarrollo de software y sus beneficios en cuanto a la mejora de procesos y calidad de productos, por lo que la recolección, el almacenamiento y el análisis de métricas se han convertido en un aspecto crítico para el éxito de un proyecto de desarrollo de software. En este sentido, el propósito de este trabajo ha sido diseñar un repositorio para almacenar y recuperar métricas de software obtenidas mediante minería de datos y construir un prototipo para evaluar la operabilidad del repositorio, basándose para ello en el modelo presentado por Luna *et al.* [1]. Así, a lo largo de las siguientes secciones se describirá el trabajo realizado, iniciando con los fundamentos teóricos sobre el modelo base, continuando con la descripción de la metodología y una semblanza del estado del arte, terminando con los resultados obtenidos y las conclusiones correspondientes.

Fundamentos teóricos

El modelo base de Luna *et al.* [1] fue desarrollado para realizar consultas en lenguaje natural, compuesto de una parte estática para almacenar las métricas y de una parte dinámica para recuperarlas. De acuerdo al modelo, la gestión de métricas es llevada a cabo por el administrador del repositorio, quien almacena en el repositorio las métricas extraídas de las bases de datos originales; las métricas extraídas son clasificadas de acuerdo a su función, permitiéndose la posibilidad de realizar subclasificaciones a cualquier nivel. De esta manera, la información extraída es ubicada en un conjunto de estructuras que permiten identificarla y almacenarla en forma de metadatos, siendo posible recuperarla mediante alias de identificadores.

Metodología

Dado que este trabajo consistió básicamente en construir un prototipo de repositorio de métricas de software, la metodología seguida en este trabajo se reduce a la ejecución secuencial de las siguientes actividades:

- Revisión del estado del arte, habiéndose considerado los trabajos más relevantes hasta el momento.
- Diseño del modelo lógico que subyace en el repositorio de métricas y del modelo de clases de objetos.
- Construcción del prototipo como una aplicación web, con diversas funciones orientadas a la gestión y consulta de métricas de software.

Estado del arte

Ante todo, es importante señalar que ninguno de los trabajos en el estado del arte considera el uso de un repositorio con estructura propia para almacenar métricas, ni tampoco considera el uso de alias para recuperarlas. Así, el trabajo de Menzies y Boetticher [2] se centra en la aplicación de la minería de datos en la planeación temprana del ciclo de vida de un proyecto. Auer *et al.* [3] proponen un método para preparar métricas de software basado en la aplicación de técnicas para visualizar datos multidimensionales provenientes de diversas fuentes. Wang *et al.* [4] discuten un modelo basado en redes neuronales para predecir la calidad del software. Mertik *et al.* [5] presentan una herramienta para minar datos, Multimethod, basada en un modelo de predicción de fallas en la construcción de software. Alcala-Fernández *et al.* [6] muestran una herramienta, denominada KEEL, concebida para evaluar los algoritmos evolutivos que se utilizan en problemas de minería de datos. Eno y Thompson [7] discuten la aplicación de algunas técnicas de minería de datos para descubrir patrones que pueden ser utilizados para transformar datos originales en conjuntos de datos sintéticos, útiles para evaluar nuevo software. Gousios y Spinellis [8] presentan una herramienta orientada a minar repositorios de software, concebida como una plataforma para el análisis de la calidad del software. Shen y Liu [9] presentan un enfoque para detectar defectos en el software basándose en el análisis de reglas de conexión, realizado mediante técnicas de minería de datos. Xie *et al.* [10] discuten la aplicación de algunos algoritmos de minería de datos a diversas tareas involucradas en el proceso de ingeniería de software, esto para mejorar la productividad y calidad del software.

Arquitectura del prototipo

En la figura 1 se muestra la arquitectura de nuestro sistema de repositorio de métricas, en la que se pueden observar dos procesos generales: (1) la minería de métricas y (2) la gestión y consulta de métricas. Así, en una primera etapa, el experto en el tema, en este caso un Ingeniero de Software o similar, realiza la extracción de métricas de las bases de proyectos de desarrollo de software mediante herramientas de minería de datos. En una segunda etapa, las métricas extraídas son almacenadas en el repositorio por el administrador del sistema de acuerdo a la clasificación hecha por el experto, de manera que éstas puedan ser consultadas por el usuario final mediante alias de identificadores, como se describirá más adelante.

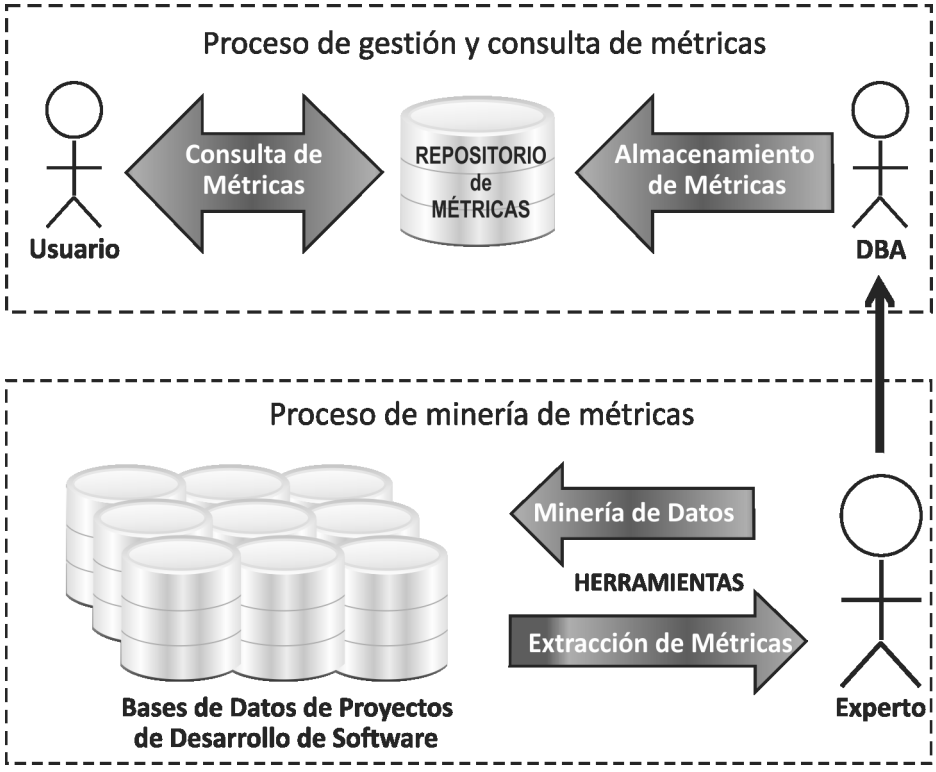

Figura 1. Arquitectura del sistema de repositorio de métricas.

Modelo de datos y modelo de objetos

En la figura 2 se muestra el modelo lógico derivado del modelo de Luna *et al.* [1], en el cual se puede observar un campo denominado RutaFTP en la tabla **Frases**, que almacena la ruta de los metadatos relacionados con una métrica específica. Es importante mencionar que aunque este modelo es la parte central del repositorio, requiere de un modelo auxiliar para poder ejecutar todas las operaciones SQL definidas para el repositorio y para poder desplegar las diferentes presentaciones definidas en la aplicación web. Tal modelo es el modelo de clases de objetos.

En la figura 3, se puede observar dicho modelo, compuesto por dos clases principales: BDconexion y Util. La primera clase es la encargada de ejecutar operaciones en lenguaje SQL, mientras que la segunda clase es la encargada del funcionamiento dinámico de la aplicación web. En la figura también se puede observar la definición de tres páginas web, *búsqueda*, *dba* e *inicio*, cuya función es controlar el acceso de los usuarios al sistema.

Así, una vez en el sistema de repositorio, la consulta de métricas es simple: se teclea cualquier alias del identificador de la métrica que se desea consultar y enseguida se desplegará una lista de métricas relacionadas. De la lista desplegada, el usuario, quien se supone un experto en el tema, podrá seleccionar aquellas métricas que sirvan a sus propósitos. De esta manera, es inmediato recuperar cualquier métrica que se desee, siempre que haya sido almacenada previamente.

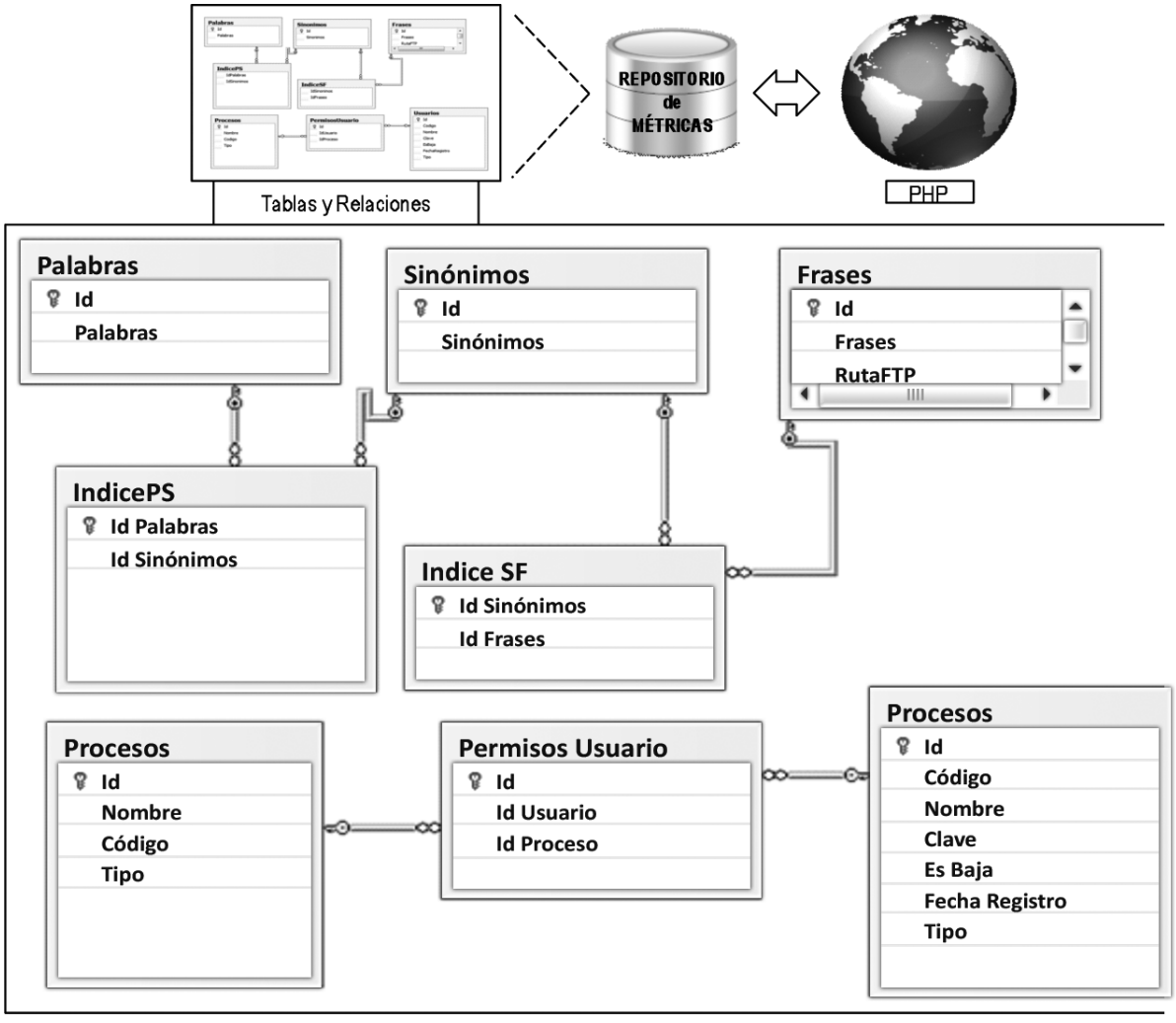

Figura 2. Modelo de datos subyacente en el repositorio de métricas de software.

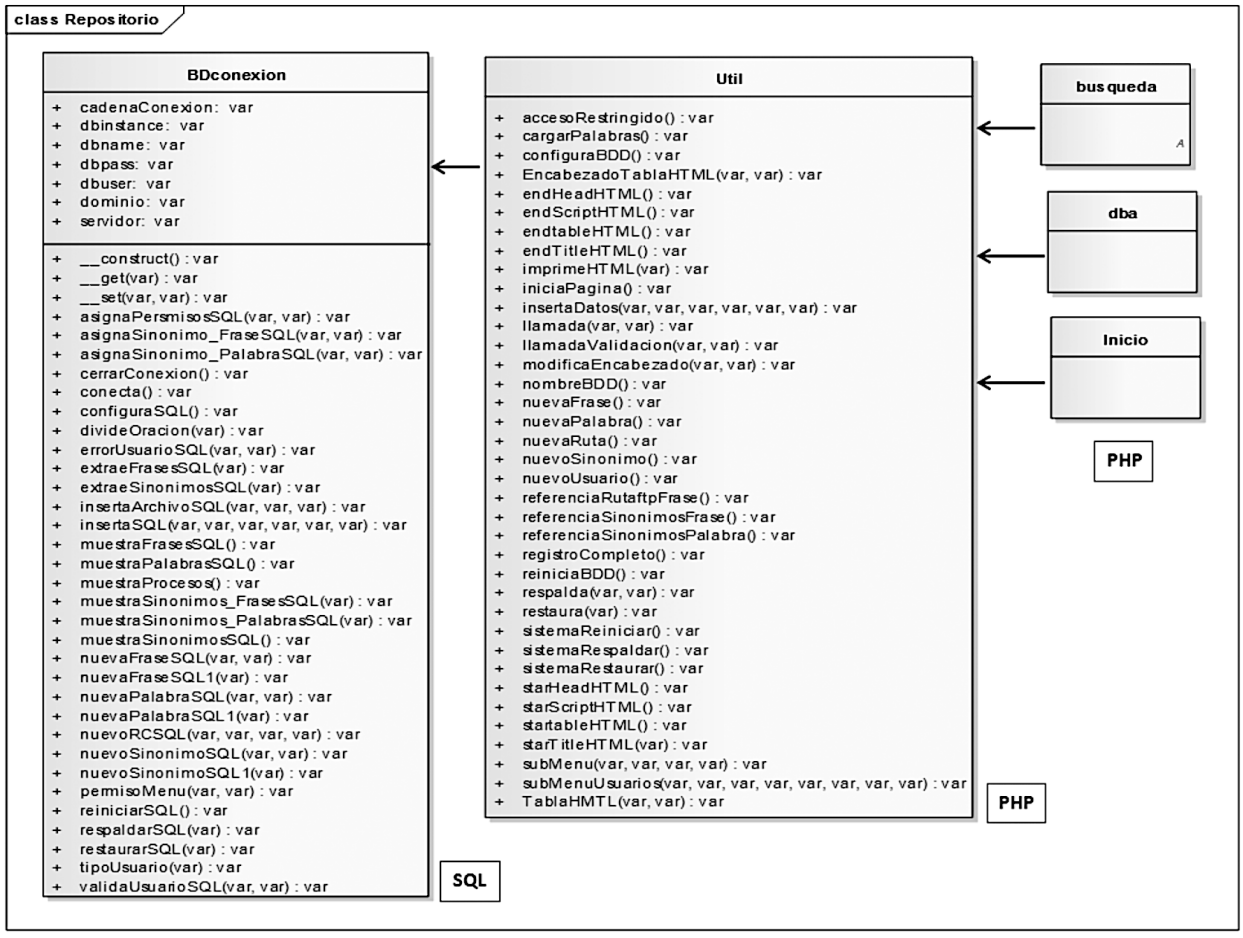

Figura 3. Modelo de clases de objetos.

Resultados

Para ilustrar lo anterior y mostrar la aplicabilidad de nuestro prototipo, en la figura 4, complementada con la figura 5, se presenta un ejemplo relacionado con la construcción de modelos de estimación del tamaño de un proyecto de desarrollo de software [11]. En este ejemplo se dispuso de datos referentes a 42 sistemas de contabilidad con características de sistemas comerciales, cada uno de los cuales incluye alguno(s) de los subsistemas de *Compras*, *Ventas*, *Inventario*, *Finanzas* y *Ciclos de Producción*. Para un estudio conveniente de estos subsistemas, se realizó la descomposición de los datos en tres grupos diferentes siguiendo las pautas de Verner y Tate [12] y se identificaron atributos en cada uno de los grupos. De esta manera, se obtuvieron en total 1537 registros con los siguientes atributos:

- TYPECOMP: Tipo de componente (1: menú, 2: entrada, 3: informe/consulta).
- OPTMENU: Número de opciones (sólo para componentes de tipo 1).
- DATAELEMEN: Número de elementos de datos (sólo para componentes de tipos 2 y 3).
- RELATION: Número de relaciones (sólo para componentes de tipos 2 y 3).
- LOC: Número de líneas de código del módulo.

Para obtener las relaciones entre estos atributos, se analizaron sus valores mediante *gráficos de dispersión* (discutidos más adelante), que permitieron detectar una clara influencia de dos atributos sobre el número de líneas de código: DATAELEMEN y RELATION.

Es importante mencionar que para contabilizar las entidades en los diferentes subsistemas, éstas fueron clasificadas en primarias, consideradas como aquellas para las que se construye el sistema con la intención de recolectar y almacenar datos, y no primarias, que son las utilizadas como referencia, validación, etc. De esta manera, se contabilizaron las entidades a las que se hace referencia en cada transacción, no el número de referencias en sí.

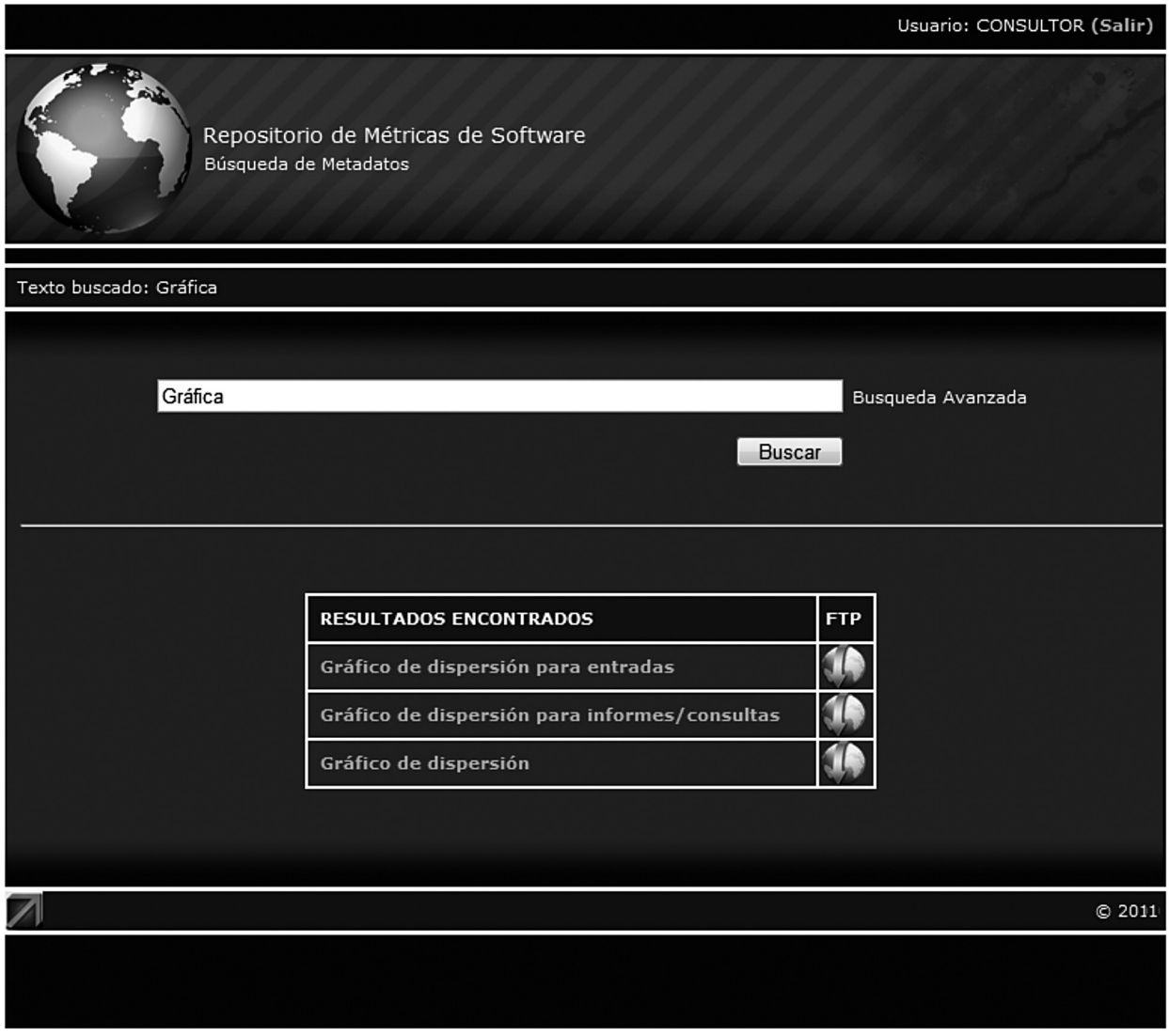

Figura 4. Búsqueda de métricas de software mediante alias de identificadores.

En la figura 4 se puede observar la consulta/búsqueda de métricas relacionadas con el tema de *Gráficos de dispersión*, la cual fue realizada mediante el alias "Gráfica" (en este caso, el alias consiste de una sola palabra, pero un alias puede ser también una frase). De la lista de resultados (identificadores) encontrados, se puede seleccionar cualquiera de ellos. En particular, se puede seleccionar "Gráfico de dispersión", en cuyo caso se obtendrá la gráfica de la figura 5, que muestra una influencia importante de los atributos RELATION y DATAELEMEN en la estimación del tamaño de un proyecto de desarrollo de software (atributo LOC), con base en el denso sombreado observado. Cabe señalar que al igual que todos los metadatos utilizados en este proyecto, esta gráfica fue obtenida mediante el uso de herramientas de minería de datos, habiéndose utilizado para este caso la herramienta MineSet [13]).

Conclusiones y trabajo futuro

En este artículo se presentó un prototipo de repositorio para almacenar y recuperar métricas de software, diseñado para consultar métricas de interés mediante alias de identificadores, lo que supone una mayor flexibilidad y eficiencia durante el proceso de consulta métricas y, por ende, un apoyo importante en el proceso de desarrollo de software.

Se mostró la factibilidad de la operación del sistema de repositorio en ambientes de trabajo relacionados con el tema, de suerte tal que en un futuro próximo se tiene considerada su implantación en dependencias gubernamentales del Estado de Aguascalientes que incluyan desarrollo de software en sus funciones.

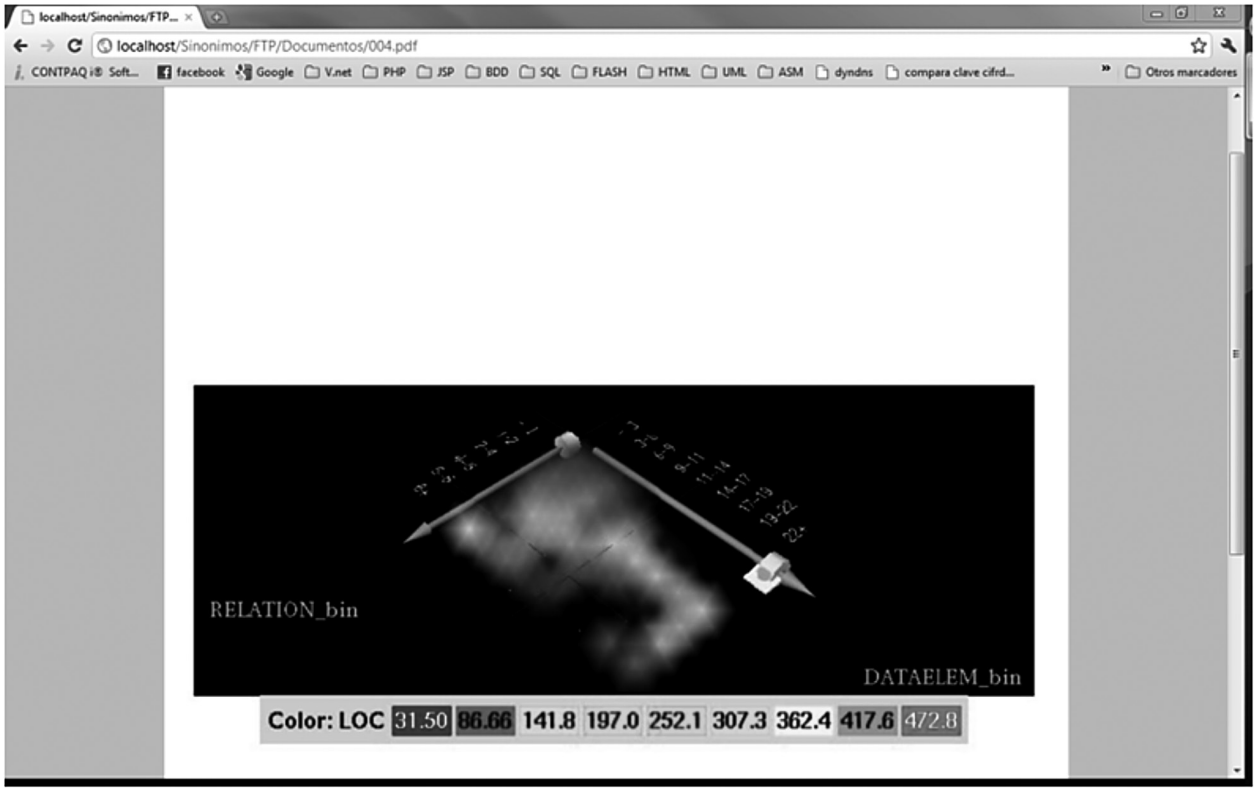

Figura 5. Metadato asociado a la métrica "Gráfico de dispersión".

También, como trabajo futuro, se está considerando la posibilidad y conveniencia de automatizar algunos procesos de gestión de métricas en el sistema de repositorio, que actualmente son realizados en forma manual por el administrador del sistema. Un ejemplo de esto es la inserción automática de métricas en el sistema de repositorio.

Finalmente, es importante señalar que el prototipo presentado no pretende competir con herramienta alguna de minería de datos, ya que su orientación es más bien a la gestión del conocimiento (métricas) extraído mediante este tipo de herramientas.

Agradecimientos: Al Cuerpo Académico "Ingeniería del Conocimiento" del *Instituto Tecnológico El Llano Aguascalientes*, a la *Dirección General de Educación Superior Tecnológica* y al Programa de Mejoramiento del Profesorado (PROMEP) de la *Subsecretaria de Educación Superior* por su apoyo a este trabajo a través del proyecto registrado con la clave ITELL-CA-4, IDCA 8314. Al Cuerpo Académico "Tecnologías de la Información y Comunicación" de la *Universidad Tecnológica de Aguascalientes* por su colaboración en este trabajo.

Referencias

[1] E. Luna, F. J. Álvarez, J. M. Espinoza, H. Ambriz y J. A. Nungaray. (2010). "Modelo para Almacenar y Recuperar Métricas de Software". *Conciencia Tecnológica*, No. 39, pp. 31-37, Enero-Junio 2010.

[2] T. Menzies & G. D. Boetticher. (2002). "Smarter software eng.: practical data mining approaches". *Proc. on Software Engineering Workshop - 27th Annual NASA Goddard*, pp. 1-38.

[3] M. Auer, B. Graser & S. Biffl. (2003). "An approach to visualizing empirical software project portfolio data using multidimensional scaling". *IEEE International Conference on Information Reuse and Integration*, pp. 504-512.

[4] Q. Wang, B. Yu & J. Zhu. (2004). "Extract rules from software quality prediction model based on neural network". *16th IEEE Int. Conference on Tools with Artificial Intelligence*, pp. 191-195.

[5] M. Mertik, M. Lenic, G. Stiglic & P. Kokol. (2006). "Estimating Soft. Quality with Advanced Data Mining Techniques". *Int. Conf. on Software Engineering Advances*, p. 19.

[6] J. Alcala-Fernández, S. García, F. J. Berlanga, A. Fernández, L. Sánchez & F. Herrera. (2008).

"KEEL: A data mining software tool integrating genetic fuzzy systems". *3rd International Workshop on Genetic and Evolving Systems*, pp. 83-88.

[7] J. Eno & C. W. Thompson. (2008). "Generating Synthetic Data to Match Data Mining Patterns". *IEEE Internet Computing*, 12:3, pp. 78-82.

[8] G. Gousios & D. Spinellis. (2009). "A platform for software engineering research". *6th IEEE International Working Conference on Mining Software Repositories*, pp. 31-40.

[9] Y. Shen & J. Liu. (2009). "Research on the Application of Data Mining in Software Testing and Defects Analysis". *2nd International Conference on Intelligent Computation Technology and Automation*, pp. 619 – 621.

[10] T. Xie, S. Thummalapenta, D. Lo & C. Liu. (2009). "Data Mining for Software Engineering". *Computer*, 42:8, pp. 55-62.

[11] J. J. Dolado. (2000). "A Validation of the Component-Based Method for Software Size Estimation". *IEEE Transactions on Software Engineering*, 26:10, pp. 1006-1021.

[12] J. Verner and G. Tate. (1992). "A Software Size Model", *IEEE Transaction of Software Engineering*, 18 (4), pp. 265-278.

[13] MineSet. http://www.the-data- mine.com/bin/ view/Software/MineSet, visitado el 30 de junio de 2011.

Artículo recibido: 10 de agosto de 2011
Aceptado para publicación: 23 de marzo de 2012