El nacimiento de una herramienta educativa libre

Jorge Ramirez, Loraine Gimson, Gustavo Gil
Centro de Investigación y Desarrollo en Informática Aplicada (CIDIA), Universidad Nacional de Salta.
{ramirezj, loraine, gdgil}@cidia.unsa.edu.ar

RESUMEN

El software libre nos brinda una oportunidad inestimable para adecuar una aplicación a los requerimientos específicos de una propuesta docente, o para proponer o iniciar nuevos desarrollos que se adecuen a esas necesidades. En este trabajo presentamos las consideraciones pedagógicas y técnicas adoptadas para comenzar el desarrollo de una nueva herramienta educativa siguiendo el proceso de desarrollo del software libre. Específicamente, se trata de construir una aplicación que permita crear mapas conceptuales sencillos y que a la vez se pueda utilizar como soporte de una presentación mediante una interfaz de zooming dirigida por el propio docente o estudiante que diseña el diagrama en cuestión. También, exponemos las características del desarrollo propuesto, el estado actual, las limitaciones y expectativas y presentamos las conclusiones provisionales obtenidas de las experiencias realizadas hasta este momento.

Introducción

La educación con TICs (Tecnologías de la Información y la Comunicación) plantea nuevos escenarios en los cuales resulta indispensable repensar la intervención del docente. Sin embargo, la incorporación de recursos tecnológicos no necesariamente implica innovación didáctica[1]; más aún, un determinado software podría imponer restricciones para el diseño de las actividades didácticas. Por ejemplo, las licencias de las aplicaciones privativas establecen frecuentemente condiciones para su utilización, pudiendo requerir la compra de licencias adicionales por cada máquina en la que se utilice. Otra limitación puede derivarse de la propia concepción con la que se desarrolló una aplicación, como ocurre con los programas tutoriales o los productos para autoaprendizaje.

Existe una enorme oferta de software diseñado con fines educativos o utilizables para la educación; sin embargo, un docente puede plantearse actividades que no concuerden con las prestaciones de los productos disponibles, o puede resultar demasiado oneroso contar con una aplicación que se adapte a sus propuestas educativas.

El software libre nos brinda una oportunidad inestimable para adecuar una aplicación a los requerimientos específicos de una propuesta docente, o para proponer o iniciar nuevos desarrollos

que se adecuen a esas necesidades.

En este trabajo presentamos las consideraciones pedagógicas y técnicas adoptadas para comenzar el desarrollo de una nueva herramienta educativa. Específicamente, se trata de construir una aplicación que permita crear mapas conceptuales sencillos y que a la vez se pueda utilizar como soporte de una presentación mediante una interfaz de zooming. En la sección siguiente repasamos diferentes trabajos relacionados con el desarrollo de software educativo afín a la aplicación que nos estamos planteando. A continuación, relevamos una serie de propuestas para orientar el desarrollo de una aplicación libre. Posteriormente, exponemos las características del desarrollo propuesto, el estado actual, las limitaciones y expectativas. Finalmente, presentamos las conclusiones de estas experiencias las cuales son necesariamente de carácter provisional

Consideraciones educativas

Ya a fines de los '90, Jonassen[2] y otros planteaban que las tecnologías deben servir más para ayudar en la construcción del conocimiento que para instruir a las personas. En los últimos años se ha publicado un número creciente de aplicaciones orientadas a facilitar la organización de conocimientos o representar visualmente relaciones y jerarquías entre conceptos; en el artículo mencionado más arriba Jonassen denomina Mindtools a los programas computacionales que, al ser usados por los estudiantes para representar sus conocimientos, los ubican en una posición crítica respecto de los contenidos que están estudiando. Entre las herramientas de este tipo, Jonassen destaca a las “herramientas de organización semántica”, donde menciona a las aplicaciones para crear mapas conceptuales.

Los mapas conceptuales surgieron en los años '70, de la mano de Joseph Novak[3], en un proyecto que buscaba comprender los cambios en los conocimientos de los niños en el área de las ciencias. Estos mapas ponen de relieve estructuras, jerarquías y relaciones entre conceptos, y se han convertido en un instrumento para la representación de conocimientos y para el aprendizaje visual[4]. Estas ideas impulsaron el desarrollo de diversas aplicaciones para trabajar con mapas conceptuales y otros diagramas para la representación y organización del conocimiento; entre ellas se destacan CmapTool [5] del Florida Institute for Human and Machine Cognition-aplicación gratuita pero no libre- y Visual Understanding Environment [6] desarrollado en la Universidad de Tufts en los Estados Unidos.

A pesar de la variedad de la oferta, no encontramos aplicaciones que permitieran compartir un mapa conceptual en forma presencial, mediante una presentación dirigida por el propio docente o estudiante que diseña el diagrama en cuestión. Las dos aplicaciones mencionadas anteriormente, VUE y CmapTools, enfatizan y brindan recursos para el trabajo colaborativo; VUE incluso prevé

que el usuario defina caminos o “pathways” que posibilitan exponer secuencias determinadas de conceptos como una presentación o una serie de diapositivas. Sin embargo, las posibilidades de exhibir aspectos puntuales de los mapas es bastante limitada (al menos hasta la versión más reciente al momento de redactar este trabajo, la 3.1.1).

Para las actividades que nos planteamos, querríamos que el expositor tenga la posibilidad de ir redirigiendo el énfasis sobre diferentes partes del diagrama. Las aplicaciones de presentaciones tradicionales (como MS PowerPoint u OpenOffice Impress) no resultan adecuadas para nuestros propósitos, ya que ofrecen formas limitadas de navegación que no preservan la estructura ni las relaciones entre las diapositivas. Estas restricciones nos sugieren utilizar zooming; ya Good y Bederson[7] propusieron las interfaces con zoom (ZUI, Zooming User Interfaces) como una forma de mostrar presentaciones que facilita la navegación hacia diapositivas (en nuestro caso, representaciones de conceptos) por fuera de la linealidad, manteniendo al mismo tiempo las vinculaciones entre los elementos; en particular, esos autores remarcan la importancia de permitir que la presentación refleje la estructura y los diferentes niveles de detalle de los contenidos.

Desarrollo de un Software Libre

Como expuso Brooks en su célebre ensayo[8], la solución más radical para la construcción de software es no construirlo de modo alguno. Dado que la aplicación que necesitamos no existe (o no conocemos ninguna que cumpla con los requerimientos planteados), nos propusimos desarrollarla a partir de proyectos ya existentes.

El desarrollo no tiene por qué partir desde cero. El Software Libre nos pone ante la oportunidad de reutilizar o partir de otras aplicaciones[9] para construir la solución que buscamos.

También resulta de importancia la posibilidad de que nuevos desarrolladores se unan al proyecto, en la medida en que la aplicación sea de utilidad para otras personas, o que surjan nuevas alternativas que la superen o complementen.

La decisión de iniciar un desarrollo de software libre se basan principalmente en:

- La disponibilidad cada vez mayor de aplicaciones libres, cuyas licencias permiten adaptarlas, modificarlas o partir de ellas para nuevos desarrollos.
- La creciente producción en investigación sobre desarrollo de y con software libre (en general, bajo denominaciones como Software Libre y de Código Abierto o Libre Software)
- La posibilidad de iniciar una experiencia de desarrollo que tome en cuenta los avances en la ingeniería del software libre y permita extraer conclusiones a partir de ella

Disponibilidad

En los repositorios de software libre (como SourceForge, Savannah, Google Code, entre otros) se pueden encontrar cientos de miles de proyectos libres y de código abierto. Sin embargo, muchos de ellos no han publicado ninguna versión o llevan mucho tiempo sin actividad alguna de desarrollo. Un estudio cuantitativo realizado por Weiss en 2005 sobre el repositorio SourceForge consideraba como proyectos discontinuados a aquellos que habían sido registrados más de un año atrás, pero que no habían cambiado su estado de "pre-alfa"; esos proyectos constituían más del 43%[10].

Se han propuesto diversos criterios para evaluar el éxito o el fracaso de un proyecto de software libre. English y Shweik[11] propusieron una clasificación en seis categorías, tomando en cuenta la etapa de desarrollo en que se encuentra el proyecto y la actividad registrada en su producción reciente.

Tampoco es fácil definir en qué consiste un proyecto exitoso de software libre; Crowston, Weiss y Michlmayr [12][13][14]-entre otros- propusieron criterios y métricas para evaluar el éxito o la madurez de un proyecto de este tipo. Sin embargo, el grado de actividad de desarrollo reciente aparece en común como un aspecto a tener en cuenta respecto de la vitalidad actual de un proyecto.

La investigación sobre el desarrollo de Software Libre:

Para orientar el desarrollo, repasamos una serie de investigaciones relevantes referidas a los procesos de desarrollo propios de software libre y las características o condiciones de ese desarrollo tendientes a obtener un producto comunitario exitoso.

Indudablemente, el ensayo "la Catedral y el Bazar" de Eric Raymond[14] constituye una referencia fundamental en la caracterización y el estudio del desarrollo de Software Libre y de Código Abierto. En ese célebre trabajo contrapone una metodología descentralizada y ágil (dicho en términos actuales) propia del software libre -a la que compara con un mercado- a la construcción sistemática y estructurada que sería propia del mundo del desarrollo industrial de software. En ese texto, Raymond extrae una serie de características beneficiosas para el proceso de desarrollo, entre las cuales se destacan:

- Un buen proyecto de software parte de la necesidad del desarrollador (o, en sus términos más coloquiales, de "rascarse una comezón")
- Publicar rápida y frecuentemente. La publicación rápida apunta a ampliar la base de interesados en el proyecto; la frecuencia busca la retroalimentación, de manera similar a las propuestas de las metodologías ágiles.
- Considerar a los usuarios como "co-desarrolladores", encargados de probar las aplicaciones e informar fallos y hacer propuestas para futuros desarrollos.

El ensayo de Raymond se basa en su experiencia personal, no en un trabajo sistemático de investigación. Estudios posteriores pusieron en tela de juicio el carácter general de la metodología esbozada por Raymond en el mundo del software libre y de código abierto; a partir de un relevamiento propio y de datos obtenidos de otras fuentes, Krishnamurti[15] observaba en 2002 que la enorme mayoría de los proyectos maduros de software libre son desarrollados por muy pocas personas, contrariamente a lo que se esperaría del modelo de bazar mencionado más arriba. Otros estudios[16][17] concluyen que los proyectos exitosos de Software Libre tienden a seguir un ciclo de vida en el que pasan por fases centralizadas (o de "catedral") y comunitarias (o de "bazar").

Senyard y Michlmayr[17] consideran que un proyecto exitoso *debe pasar de la fase de catedral a la de bazar* a partir de la publicación del código fuente y la incorporación de usuarios y desarrolladores al proyecto. Entre los factores de éxito propuestos por estos autores se destacan:

- Un prototipo funcional promisorio, que permita a los usuarios ejecutarlo y darse una idea de las funciones buscadas
- Diseño modular en el proyecto inicial, para permitir el trabajo simultáneo de los diferentes desarrolladores, mejorar la mantenibilidad y la evolutividad
- Mecanismos adecuados para la comunicación y la contribución por parte de usuarios y desarrolladores
- Estilos o estándares de codificación claramente definidos
- Ciclos de publicación breves que liberen versiones funcionales y estables (siguiendo la idea de Raymond de publicar rápida y frecuentemente)
- Documentación clara y apropiada para desarrolladores y usuarios
- Una licencia de distribución atractiva, tanto para desarrolladores como para usuarios

Fogel[18] también presenta una serie de ideas para desarrollar software libre exitoso, enfatizando los errores comunes en este tipo de proyectos. Al igual que los autores mencionados antes, destaca la importancia de que el desarrollo se relacione con las necesidades e intereses de los iniciadores del proyecto y de la incorporación de desarrolladores y usuarios. Este autor también enfatiza la importancia de la "apariencia" de la aplicación, aspecto habitualmente desatendido por los desarrolladores.

Las características propias del software libre y de código abierto, en particular las licencias que adoptan, apuntan a la posibilidad de reutilizar código existente y también a favorecer nuevos desarrollos. Esta cualidad requiere, como ya advertían Samoladas y otros[19], que se tomen previsiones para alcanzar la máxima mantenibilidad posible. Esos autores evaluaron un conjunto de

productos F/OSS utilizando el Índice de Mantenibilidad propuesto por el Instituto de Ingeniería de Software (SEI); si bien no existe un consenso absoluto respecto de la utilidad de esta medida[20] [21], el trabajo pone de relieve la importancia de apelar a métodos y técnicas de la ingeniería de software tradicional en el ámbito del Software Libre.

Experiencia de desarrollo

Para comenzar con el desarrollo de la herramienta, tomamos en cuenta las consideraciones del punto anterior. Los requerimientos iniciales se establecieron de manera general, siguiendo el modelo de "lista de deseos"[18][22], con la perspectiva de incorporar y refinarlos en la medida en que el proyecto crezca. La lista inicial de características del software surge de la función educativa que se pretende atender:

- Multiplataforma, para que todos los integrantes de la comunidad educativa puedan utilizarlo
- Creación visual de mapas conceptuales
- Zooming sobre cada nodo
- Almacenamiento en XML, por su carácter estándar

El proyecto SiZoop (Simple Zooming Presentations) fue registrado en SourceForge (el mayor repositorio de software libre y de código abierto) el 18 de enero de 2011.

Elegimos el lenguaje Java por ser multiplataforma y por contar dentro de la Universidad con potenciales desarrolladores capacitados en la programación con ese lenguaje.

Prototipo funcional: La primera versión publicada contaba con la funcionalidad básica de permitir la creación de mapas conceptuales simples. Los lanzamientos siguientes fueron agregando la posibilidad de incluir imágenes relacionadas con cada concepto, modificacióntipografía y color de fondo, y modificación básica de cada nodo. Desde el tercer lanzamiento la interfaz del programa está en inglés y en castellano.

Lanzamientos rápidos y frecuentes: se han publicado 5 versiones desde su registración hasta mayo de 2011.

Reutilización de software: buscamos componentes que pudieran resolver la parte de Zooming y el almacenamiento en XML.

En primer lugar buscamos frameworks libres en java que brindaran la posibilidad de crear ZUI. La búsqueda se basó en los repositorios java-source.net y sourceforge; al mismo tiempo, se utilizó un buscador Web para relevar otras aplicaciones que pudieran no estar incluidas en los repositorios mencionados.

Analizamos dos alternativas: Prefuse Information Visualization Toolkit y Piccolo 2D. El primero de ellos está orientado a ofrecer distintas formas de visualizar información, en tanto que el segundo es propiamente un framework. Se descartó el primero fundamentalmente por el tiempo que lleva sin actividad de desarrollo: la última publicación data de octubre de 2007 (http://prefuse.org).

Piccolo2D, en cambio, es específicamente un framework para desarrollar aplicaciones con ZUI. La última versión lanzada es del 14 de abril de 2011. Otro aspecto destacable es la cantidad de aplicaciones que utilizan este framework (http://piccolo2d.org)

Para la tarea de almacenar en formato XML, analizamos JiBX y XOM. El primero es más flexible, pero requiere de un paso intermedio para incorporar el contenido XML a clases java. Por lo tanto, se adoptó la segunda.

Herramientas de comunicación y de gestión de proyectos: SiZoop está alojado en SourceForge, principalmente por las posibilidades que ofrece en cuanto a gestión de proyectos y comunicaciones. Se dispone de un servidor SVN, espacio Web propio para la aplicación, herramientas de seguimientos de errores, foros, etc.

Política de aseguramiento de la calidad: en el desarrollo llevamos el registro de los valores de un conjunto de métricas (provisto por el plugin Eclipse Metrics) con el fin de mantener el seguimiento sobre distintas porciones del código y detectar tempranamente posibles riesgos para la mantenibilidad. Por ejemplo, en la versión 0.5 se observan dos módulos cuyo valor de complejidad ciclomática supera a los umbrales propuestos por McCabe (ver ilustración 1). Tal valor representa sólo una parte de la complejidad[23] por lo que estos dos módulos fueron sometidos a revisión, comprobándose que el valor computado no refleja mayor dificultad para comprender el código.

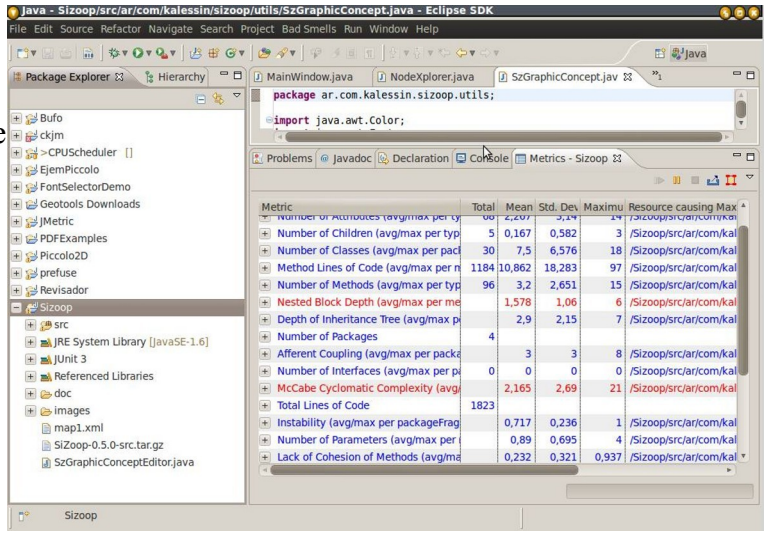

Ilustración 1: Vistazo de EclipseMetrics en el desarrollo de SiZoop.

Licencia: SiZoop se distribuye bajo una licencia dual (GPL y BSD) para brindar opciones diferentes a potenciales desarrolladores, al mismo tiempo atender a las diferencias entre las licencias de los componentes reutilizados (LGPL para XOM, BSD para Piccolo2D).

Documentación: dado que el proyecto aún es de tamaño pequeño (menos de 2K líneas de código nuevas), la única documentación existente del desarrollo es el JavaDoc generado durante la

codificación.

Estado actual del desarrollo: al momento de redactarse este trabajo, como mencionamos anteriormente, se han publicado 5 versiones de la aplicación. En http://sourceforge.net/projects/sizoop están publicados los archivos binarios (jar) de todas las versiones y el código fuente correspondiente a las versiones 0.3 y 0.5. Bajo SVN, en tanto, está disponible toda la evolución del código.

La última versión publicada a la fecha consta de 1823 líneas de código; cabe mencionar que la versión utilizada de Piccolo2d tiene 7.341 líneas de código mientras que XOM consta de 47732 LOC.

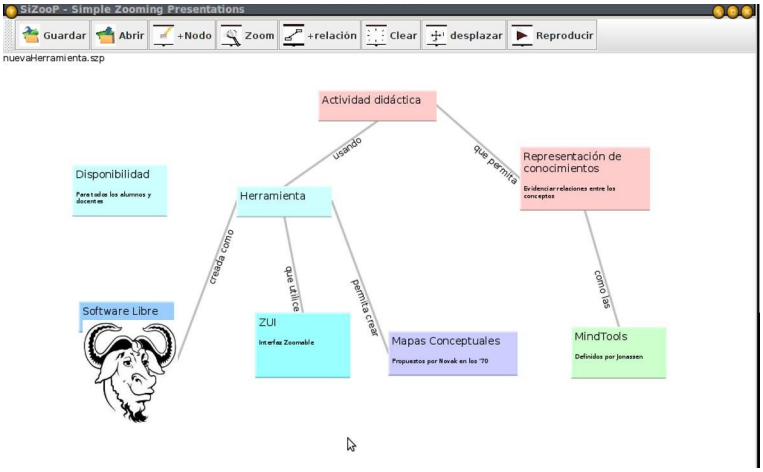

Ilustración 2: SiZoop 0.5 en funcionamiento

Conclusiones y futuros trabajos

Este trabajo presentamos los fundamentos educativos que orientan la creación de una nueva aplicación informática y proponemos el desarrollo de un software Libre para atender esa necesidad.

El desarrollo de software libre surge frecuentemente de las propias necesidades de los desarrolladores. Sin embargo, desde el momento en que la aplicación y su código fuente se ponen a disposición del público, se promueve la posibilidad de que el producto resulte de utilidad para otros actores. Estos, a su vez, podrán introducir mejoras o incluso generar nuevas iniciativas que -eventualmente- resulten superadoras del producto inicial.

El Software Libre también permite partir de aplicaciones preexistentes para realizar nuevos desarrollos. La aplicación SiZoop reutiliza el framework Piccolo2D y la librería XOM, gracias a las cuales logra presentar una funcionalidad básica con menos de 2K líneas de código (1823 SLOC, medido con SLOCCount)

El desarrollo fue encarado tomando en consideración el estado del arte en cuanto a la investigación

sobre los procesos de desarrollo en el ámbito del software libre y de código abierto.

En la medida en que la aplicación resulte de interés para otras personas, el proyecto tendrá posibilidades de prosperar. Sin embargo, la posibilidad de que SiZoop se adecue a las necesidades educativas de otros docentes, no garantiza el éxito ni la evolución del proyecto; el equipo que inicia el desarrollo tendrá la responsabilidad de mantener una gestión cuidadosa, incrementar y mejorar la documentación, a la espera de que el propio proyecto trascienda más allá del grupo de origen; si alcanza el estadio de "bazar", constituirá una experiencia valiosa de desarrollo de software libre y demostrará utilidad para la comunidad.

Bibliografía:

[1] Adell, J.:Riesgos y Posibilidades de las TICs en educación. http://webquest.xtec.cat/articles/adell_bernabe/2006/riesgosyposibilidades.pdf.
[2] Jonassen, D.H.; Carr,C.; Yueh, H.: «Computers as Mindtools for Engaging Learners in Critical Thinking», *TechTrends*, Assosiation for Educational Communications & Technology,1998.
[3] Novak, J.; Cañas, A.:Origen y desarrollo de los mapas conceptuales. http://cmap.ihmc.us/docs/origenes.html.
[4] Jonassen, D.; Marra, R.: «Concept mapping and other formalisms as mindtools for representing knowledge », *Research in Learning Technology*, Association for Learning Technology,1994.
[5] Cañas, A.; Hill,G.; Carff,R.; Suri,N.; Lott,J.; Gómez,G.; Eskrisge,T.; Arroyo,M.; Carvajal, R.. *CMapTools: A knowledge Model and sharing environment*. En *First International Conference on Concept Mapping*. 2004.
[6] Kumar, A.; Kahle, D.. *VUE: a concept mapping tool for digital content*. En *Second International Conference on Concept Mapping*. 2006.
[7] Good, L.; Bederson, B.B.: «Zoomable user interfaces as a medium for slide show presentations», *Information Visualization*, Palgrave Macmillan,2002.
[8] Brooks, F.P.: «No Silver Bullet Essence and Accidents of Software Engineering», *Computer*, IEEE Computer Society Press,1987.
[9]Brown, A.; Booch, G.: «Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors», en *Software Reuse: Methods, Techniques, and Tools*, SpringerLink, 2002.
[10] Weiss, D.. *Quantitative Analysis of Open Source Software Projects on SourceForge*. En *1st International Conference on Open Source Software System*. 2005.
[11] English, R.; Schweik, C.: «Identifying Success and Abandonment of FLOSS Commons: A Classification of Sourceforge.net Projects», *UPGRADE: The european Journal for the informatics Professionals*, ,2007.
[12] Crowston, K.; Annabi,H.; Howison, J.. *Defining Open Source Software Project Success*. En *in Proceedings of the 24th International Conference on Information Systems (ICIS 2003)*. 2003.
[13] Weiss, D.. *Measuring success of open source projects using web search engines*. En *First International Conference on Open Source Systems*. 2005.
[14] Michlmayr, M.. *Software Process Maturity and the Success of Free Software Projects*. En *Proceeding of the 2005 conference on Software Engineering: Evolution and Emerging*

Technologies. 2005.
[15] Krishnamurthy, S.: «Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects», *SSRN eLibrary*, SSRN,2002.
[16] Capiluppi, A.; Michlmayr, M.. *From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects*. En *Open Source Development, Adoption and Innovation*. 2007.
[17] Senyard, A.; Michlmayr, M.. *How to Have a Successful Free Software Project*. En *Proceedings of the 11th Asia-Pacific Software Engineering Conference*. 2004.
[18] Fogel, K.: *Producing Open Source Software: How to Run a Successful Free Software Project*, O'Reilly Media, Inc., 2005.
[19] Samoladas, I.; Stamelos,I.; Angelis,L.; Oikonomou, A.: «Open source software development should strive for even greater code maintainability», *Commun. ACM*, ACM,2004.
[20] Heitlager, I.; Kuipers,T.; Visser, J.: «A Practical Model for Measuring Maintainability», *Quality of Information and Communications Technology, International Conference on the*, IEEE Computer Society,2007.
[21] Welker, K.: «The Software Maintainability Index Revisited», *CrossTalk, the Journal of Defense Software Engineering*, ,2001.
[22] Scacchi, W.: «Understanding the requirements for developing open source software systems», *IEE Proceedings - Software*, ,2002.
[23] Fenton, N.E.; Pfleeger, S.L.: *Software Metrics: A Rigorous and Practical Approach*, PWS Publishing Co., 1998.