ISSN0121-1129

Herramientas libres para modelar *software*

Free tools to model *software*

Mauro Callejas Cuervo*
Oscar Yovany Baquero Moreno**

RESUMEN

Observación acerca del *software* libre y de su implicación en procesos de desarrollo de *software* con herramientas 4G por parte de entidades o personas sin capitales astronómicos y sin la mentalidad *acaparadora* de dominar el mercado con productos costosos que las hagan multimillonarias y que no ofrecen una garantía real, ni la posibilidad siquiera de conocer el *software* por el que se ha pagado, y mucho menos de modificarlo si no cumple nuestras expectativas.

Palabras clave: Software libre, GPL, Diagramas dinámicos, Diagramas estáticos, UML.

ABSTRACT

An observation about the free software and its implication in *software* development processes, with 4G tools by entities or people without astronomical capitals and without the *monopolistic mentality,* which intends to dominate the market with expensive products in order to become multimillionaire, without a real guarantee, neither possibility to know the software for which one has been paid for, and much less to modify it, if it does not full fills one's expectations.

Key words: Free software, GPL, Dynamic diagrams, Static diagrams, UML.

* Ingeniero de Sistemas, Especialista en Ingeniería de Software, Tesista de Maestría en Ciencias Computacionales. Profesor Escuela de Sistemas y Computación de la UPTC. Tunja, Boyacá, Colombia, c.e.: maurocallejas@yahoo.com.
** Estudiante Ingeniería de Sistemas y Computación. Escuela de Sistemas y Computación UPTC, c.e.: oybm@hotmail.com.-.

INTRODUCCIÓN

Si como seres humanos hemos buscado a través de los tiempos liberarnos de nuestros opresores físicos, morales, religiosos, ¿por qué no buscar la libertad intelectual?

Un punto particularmente llamativo del *software* propietario es que, en la mayoría de los casos, el comprador no puede compartir su compra con un tercero. Por otra parte, la empresa desarrolladora se deshace de toda responsabilidad respecto a las pérdidas que pudiera ocasionar para el comprador el uso del *software;* esto incluye los daños por fallas del producto, debidas a sus errores. En este sentido, la licencia aclara que el vendedor no garantiza la aplicabilidad del producto a ningún fin determinado.

En general, se presentan tres desventajas básicas al adquirir *software* propietario: dependencia de un proveedor, falta de garantías y falta de soporte, cuando los requerimientos han variado un poco.

Y la pegunta frecuente: ¿qué es *software* libre? Un *software* puede considerarse «Libre'' si, argumenta Richard Stallman, se distribuye bajo una licencia que garantice a quien lo recibe las siguientes libertades [1]:

- Libertad de ejecutar el programa, con cualquier propósito.
- Libertad de estudiar cómo funciona el programa y de adaptarlo a sus necesidades.
- Libertad de redistribuir copias del programa.
- Libertad de mejorar el programa y redistribuir dichas modificaciones.

Estos últimos dos puntos deben tener una restricción: si redistribuye el programa, modificado o no, debe hacerse bajo las mismas condiciones. Esta cláusula es la que garantiza la libertad del programa, impidiendo que alguien tome un desarrollo libre y lo transforme en un desarrollo propietario, limitando alguna de las libertades citadas.

El desarrollo de *software* ha cambiado y ha pasado a ser un compendio de aspectos y herramientas innovadores utilizados en comunidad. En otras épocas, si el programador pretendía plasmar su idea de *software* en algún diseño que lo orientara a través del proceso de programación, debía diseñar un diagrama de flujo [2], que si en ese tiempo era útil, no era lo suficientemente poderoso para modelar muchos de los aspectos que se deben tener en cuenta para desarrollar *software*.

De la misma manera como se han creado nuevas formas de programación, por ejemplo, la orientada a Objetos [3], han cambiado las formas de plasmar las ideas en diagramas que terminan por convertirse en un proyecto de *software*.

Cierto día un grupo de caballeros, con ideas que revolucionaron separadamente en su momento la forma de diseñar un sistema, se juntaron para crear un estándar, que hoy en día es el más conocido y utilizado. Nacía UML [4, 5] por sus siglas en ingles (Lenguaje Unificado de Modelado). Si bien UML es utilizado mayormente para el diseño de sistemas de software, cabe aclarar que también es utilizado para diseñar cualquier otro tipo de sistema, ya sea electrónico, mecánico, educativo, etc.

UML cuenta con varios tipos de modelos, que permiten a los creadores de *software* plasmar sistemas con diferentes puntos de vista; su principal división da origen a la vista dinámica y la vista estática, cada una de ellas integrada por variados tipos de modelos que ofrecen una apreciación diferente del sistema.

Otro de los conceptos por tener en cuenta son las herramientas 4G y 5G (de cuarta y quinta generación) [6]; ellas permiten, además de diseñar de una forma eficiente los diagramas, generar los objetos que se convertirán paso a

paso en el *software* que se desea; si bien se deben modificar algunas líneas de código para cambiar detalles que no sean de plena satisfacción del usuario, la mayor parte del código se genera a través del diseño de diagramas que el programador realiza y la herramienta de diseño convierte a código puro.

1. RESULTADOS

Actualmente en el mercado existen diversas herramientas fabricadas para dar funcionalidad y realismo a los diagramas que un programador diseña como base fundamental del desarrollo de *software*. A continuación se analizan y comparan varias herramientas, con el fin de dar una mayor visión para la elección de herramientas de diseño que están dentro de las *open source,* adaptables a cualquier necesidad.

ArgoUML [7, 8]. Es una herramienta de modelado desarrollada en Java, que ayuda a elaborar proyectos que usan UML. ArgoUML es autorizada bajo la licencia de BSD [9], que también permite mercantilizar las extensiones. La compañía *Gentleware* ofrece una extensión bajo el nombre *Poseidon* para UML. Una edición para la comunidad se ofrece gratis. *ArgoUML* no es sólo una herramienta libre, también es un proyecto de desarrollo de *software* abierto, donde se invita a contribuir. *ArgoUML* permite elaborar diagramas de casos de uso, clases, estados, actividades y colaboraciones.

Dia [10, 11]. Es una herramienta de creación de diagramas autorizada bajo la licencia GPL (General Public License) [12]. *Dia* es *software* libre; comparado con el programa propietario «*Visio*», puede usarse para dibujar muchos tipos diferentes de diagramas. Tiene los objetos especiales para ayudar a dibujar la relación entre entidades, permite hacer diagramas de UML, diagramas de flujo, diagramas de red. También es posible implementar nuevas figuras, diseñándolas en un archivo XML [13].

Omondo EclipseUML [14]. Es una herramienta de modelado visual, originalmente integrada con *Eclipse* y *CVS*. Cuenta con una configuración para trabajo en equipo, que es capaz de administrar centenares de conexiones simultáneas y capaces de adaptarse a equipos de desarrollo de *software* grandes. La filosofía de *Omondo* [15] es integrarse completamente con *Eclipse*, respetar las normas oficiales, ser independiente de opciones tecnológicas y usar UML para integrar y contribuir al mejoramiento de las tecnologías *open source* o código abierto.

Poseidon UML. Es una herramienta realizada por *Gentleware* [16], que permite crear diagramas UML y soporta todos los diagramas de este estándar y diferentes lenguajes. Además existen diferentes versiones, algunas de las cuales, como la *community edition,* son de libre descarga. Poseidon es capaz de almacenar los diagramas en formato UML 2.0, ofrece soporte completo de undo y redo, soporte de Java, C#, VB.net, IDL, SQL DDL, Perl y Delphi, estabilidad y rendimiento mejorados, totalmente implementados en Java, independiente de la plataforma, soporte para todos los diagramas de UML, soporte para XML 1.2, formato estándar de almacenado. XML 1.0, 1.1 y 1.2 son admitidos, exporta los diagramas como gif, ps, eps y svg, soporte para los formatos jpeg y png para JDK 1.4., Copy/cut/paste en la herramienta, Drag and drop en la herramienta, internacionalización para inglés, alemán, ruso, francés, español, y chino, generación de código Java, fácil instalación de actualizaciones mediante Java Web Stara.

UMLet. Herramienta para modelado rápido de UML, también escrita en Java. *UMLet* le permite elaborar diagramas rápidamente; los diagramas pueden exportarse a eps, pdf, jpg, svg y portapapeles del sistema; permite realizar la mayoría de diagramas que usa *Eclipse* y crea sus propios elementos gráficos personalizados. Esta herramienta está disponible en el sitio web:

http://www.auer-mayr.com/umlet_free_uml_editor/index.htm.

Together [17]. Es una herramienta de código abierto desarrollada en Java, muy útil para elaborar diagramas UML rápidamente; permite algo más que dibujar diagramas bonitos. Together permite tener la certeza de que todos los diagramas UML que se realizan tienen las especificaciones, los detalles y las conductas, y genera el código fuente real y sintácticamente correcto de los objetos que están modelados en el diagrama. Finalmente, se puede realizar una compilación de la aplicación generada, mas no de los diagramas.

VisualParadigm. Esta herramienta acelera el desarrollo de aplicaciones, ya que sirve de puente visual entre arquitectos, analistas y diseñadores de sistemas de información, haciendo el trabajo más fácil y dinámico. El paradigma Visual SDE [18] para NetBeans [19] se diseña para aumentar la velocidad en el análisis, la captura, plan, desarrollo, comprobación y despliegue de los requisitos. La herramienta también automatiza tareas tediosas que pueden distraer a diseñadores del desarrollo. Ofrece los siguientes beneficios:

- La navegación intuitiva entre el código y el modelo visual.

Cuadro resumen de características

Nombre: Se integra con: Descripción: Licencia:	**Visual Paradigm SDE** NetBeans y Eclipse Permite hacer diagramas UML, generación automática de código, sincronización entre modelos y código entre otras posibilidades. Comercial, con versión *free* para la comunidad y versiones de evaluación del resto.
Nombre: Se integra con: Descripción: Licencia:	**Omondo** Eclipse Permite hacer diagramas UML, generación automática de código, sincronización entre modelos y código entre otras posibilidades. Comercial, con versión de evaluación.
Nombre: Se integra con: Descripción: Licencia:	**Together para Eclipse** Eclipse y versión standalone Permite hacer diagramas UML, generación automática de código, sincronización entre modelos y código entre otras posibilidades. Comercial, con versión de evaluación.
Nombre: Se integra con: Descripción: Licencia:	**Slime UML** Eclipse Soporte completo para casos de uso, diagramas de clases y packages. Shareware
Nombre: Se integra con: Descripción: Licencia:	**Poseidon UML** NetBeans Permite modelar con varios diagramas UML. Comercial con versión para la comunidad.

- Poderoso generador de informes PDF/HTML.

- Tiempo real en la demanda.

- Un ambiente modelador visual superior.

- Sofisticado diseño de diagramas.

2. DISCUSIÓN Y CONCLUSIONES

Las herramientas de modelado UML se convierten en partes fundamentales en el desarrollo de todo sistema de información *orientado a objetos*, ya que son las encargadas de plasmar los requerimientos del sistema, la funcionalidad y la interacción con otros módulos o sistemas.

Cuando se trata de elegir la herramienta adecuada para modelar sistemas de información basados en UML, surge una pregunta: ¿qué se debe tener en cuenta? Básicamente son tres razones principales por las cuales una empresa productora de *software* podría inclinarse por este modelo:

1. Al optar por el modelo de desarrollo y distribución libre, la empresa queda habilitada a utilizar la gran cantidad de herramientas libres disponibles en la actualidad. Esto no solo implica la ejecución de dichas herramientas, sino también su modificación, para adaptarlas a casos particulares, y la exploración de sus mecanismos de funcionamiento, para reutilizarlos en futuros desarrollos. De esta forma se obtiene una ventaja significativa frente a aquellas empresas que al basar su negocio en el modelo propietario o cerrado no pueden utilizar esta base de herramientas y conocimiento.

2. Liberar un programa facilita enormemente su distribución y publicidad. De esta manera no es necesario invertir enormes sumas de dinero en campañas publicitarias y en marketing, para poder competir con productos establecidos en el mercado.

3. Si el producto en cuestión tiene suficientes méritos técnicos, con seguridad despertará el interés de un gran número de desarrolladores, usuarios y empresas en todo el mundo, que comenzarán a contribuir en su desarrollo, extensión y depuración. Muchos son los casos en que pequeños emprendimientos han engendrado productos de gran nivel técnico y de una envergadura impensada por sus creadores originales. Basta citar, a modo de ejemplo, productos como el sistema operativo *Linux*, el servidor web *Apache*, el manejador de base de datos *MySQL*, entre otros.

3. REFERENCIAS

[1] Fundación para el Software Libre – Richard Stallman, URL: http://www.gnu.org/philosophy/free-sw.es.html, [fecha de visita: noviembre de 2004].

[2] R. Pressman: *Ingeniería del Software, un enfoque práctico*, Ed. MacGraw-Hill, 2002.

[3] B. Meyer: *Construcción de software orientado a objetos*, Ed. Prentice-Hall, 1998.

[4] I. Jacobson, G. Booch, J. Rumbaugh: *El lenguaje unificado de Modelado.* Madrid: Ed. Addison Wesley, 1999.

[5] C. Larman: *Applying UML and Patterns*, Ed. Prentice Hall, 2002.

[6] Course Technology, *Diccionario de Informática e Internet*, Ed. Thomson Course Technology, 2004.

[7] B. Reza, P. Lindsay, S. Ditlinger: *Mobile Uml*, Cambridge University Press, pág. 101, 2004.

[8] Collabnet, URL: http://argouml.tigris.org/, [fecha de visita: septiembre de 2004].

[9] Opensource.org, URL: http://opensource.org/licenses/bsd-license.php [fecha de visita: noviembre 2004].

[10] Wikipedia.org, URL: http://es.wikipedia.org/wiki/Dia_%28 programa%29, [fecha de visita: junio 2004].

[11] Gnome.org, URL: http://www.gnome.org/projects/dia/, [fecha de visita: julio 2004].

[12] Alicia S. Clark, Eduardo H. Clark: *English-to-Spanish Computer and Internet Dictionary*, Ed. Universal Publishers, 2004.

[13] Wikipedia.org, URL: http://es.wikipedia.org/wiki/XML, [fecha de visita: julio 2004].

[14] Frank Budinsky, David Steinberg, Raymond Ellersick, Ed. Merks, Stephen A. Brodsky, Timothy J. Grose, *Eclipse Modeling Framework*, Ed. Addison-Wesley Professional, 2003.

[15] Omondo, URL: http://www.omondo.com/, [fecha de visita: octubre 2004].

[16] Gentleware AG, URL: http://gentleware.com/index.php, [fecha de visita: noviembre de 2004].

[17] Borland Corp., URL: http://www.borland.com/us/products/ together/index.html, [fecha de visita: noviembre 2004].

[18] Jack J. Woehr, Vaughn Spurlin, Simeon Greene, Jesse Glick, Tim Boudreau, *NetBeans The Definitive Guide*, Ed. O'Reilly, 2002.

[19] Visual Paradigm Corp., URL http://www.visual-paradigm.com/, [fecha de visita: noviembre 2004].

4. BIBLIOGRAFÍA RECOMENDADA

Deitel y Deitel: *Cómo Programar en Java*. México: Prentice-Hall. 1995, 1056 pp.

Senn, James A.: *Análisis y Diseño de Sistemas de Información*. Segunda edición. McGraw-Hill. 1992, 333 pp.

Fowler, Martín. *UML Gota a Gota*. Addison Wesley Vongman. 1999.

Brackett, Jhon W.: *Software Requiriments*. Software Engineering Institute Education Program- Carnegie Mellon University. 1990.

Fecha de recepción: 12 de noviembre de 2004
Fecha de aprobación: 2 de diciembre de 2005