

El proyecto GIMP

Jesús Moreno León
j.moreno1@gmail.com

© Jesús Moreno León, Mayo de 2008

**Este documento se distribuye bajo la licencia “Shared-ALike”
2.5 de Creative Commons, disponible en:
<http://creativecommons.org/licenses/by-sa/2.5/>
*Siéntase libre de compartir, distribuir y adaptar este trabajo,
siempre que se mantenga esta nota con su autor y se
mantenga la misma licencia.***

Índice de contenido

El proyecto GIMP.....	1
1. Introducción.....	4
1.1. Un poco de historia.....	4
2. Licencias.....	6
3. Modelos de negocio.....	7
4. Motivación de los desarrolladores.....	8
5. Reglas de gobierno.....	11
6. Historia de desarrollo.....	14
6.1. Análisis del código fuente.....	14
6.2. Análisis del repositorio.....	16
7. Comunidad.....	19
8. Prácticas y procedimientos de desarrollo.....	21
9. Bibliografía.....	22
Apéndice A.....	23

1. Introducción

GIMP es el acrónimo de GNU Image Manipulation Program. Es un programa libre apropiado para tareas como retoque fotográfico y composición y edición de imagen.

GIMP es completamente extensible. Se ha diseñado para que puedan incorporarse extensiones y plug-ins que aumenten su funcionalidad. Su interfaz de script permite que para cualquier tarea se pueda crear un script y automatizarla.

GIMP está escrito y desarrollado bajo X11 en plataformas Unix. Pero básicamente el mismo código también funciona en Ms Windows y Mac OS X.

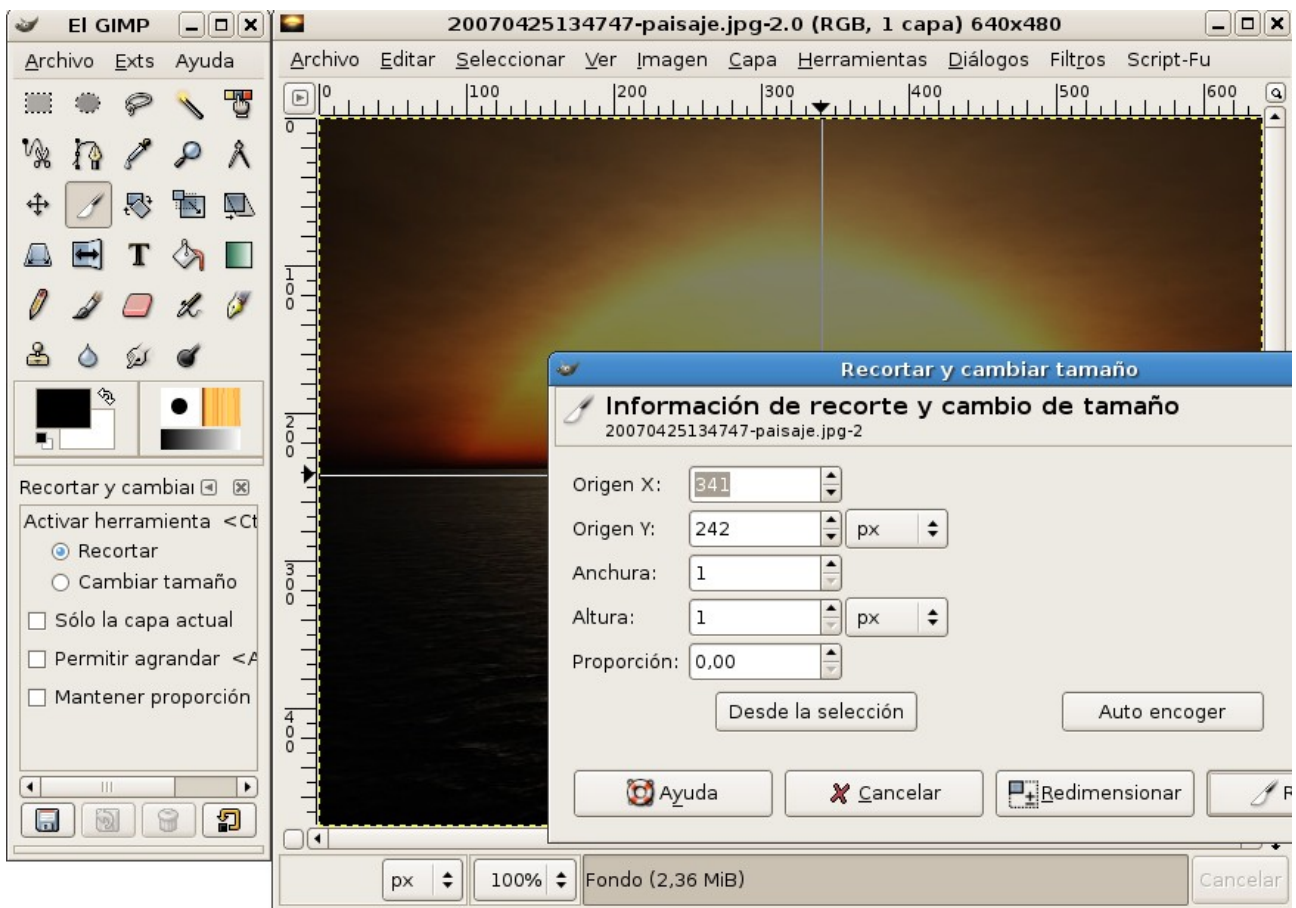


Figura 1. Captura de GIMP en acción

1.1. Un poco de historia

Un par de estudiantes de Berkeley, Peter Mattis y Spencer Kimball, decidieron en 1995 que querían escribir un programa para manipular imágenes en lugar de un compilador para una asignatura de la Universidad.

Así que se pusieron a trabajar en el programa General Image Manipulation Program, The GIMP y en Enero de 1996 lanzaron la primera release pública, la versión 0.54. Protegieron el programa con la licencia GPL y tuvo bastante éxito.

Se creó una lista de correo para desarrolladores que permitió a gente con intereses similares comunicarse sobre el proyecto. Y pronto esa lista se dividió en 2: una lista de usuarios de GIMP y otra de desarrolladores.

En seguida comenzaron a nacer sitios web, a parte del sitio principal de Berkeley, que se dedicaban a explicar cómo se usaba la herramienta. Se escribieron manuales de usuario, se creaban obras con GIMP (por ejemplo Larry Ewin creó el famoso pingüino Tux con GIMP 0.54, <http://www.isc.tamu.edu/~lewing/linux/>).

A Peter no le gustaba nada trabajar con Motif, así que desarrolló GTK (Gimp Tool Kit) para crear la interfaz de usuario de GIMP. También hubo un cambio de nombre tras una visita de Richard Stallman a Berkeley, y GIMP se convirtió en GNU Image Manipulation Program.

En Febrero de 1997 se lanzó la versión 0.99. En Junio se publicó la versión 0.99.10, la última release de los creadores originales. Peter y Spencer se graduaron, comenzaron a trabajar y abandonaron el proyecto (sin decir nada a nadie).

Tras una temporada difícil y confusa, nuevos desarrolladores dieron un paso adelante y siguieron trabajando en el proyecto, haciendo de GIMP lo que es hoy, más de 10 años después, uno de los mejores y más populares programas del mundo del Software Libre.

2. Licencias.

GIMP se distribuye con licencia GPL versión 2, en el fichero LICENSE incluido en la distribución se puede leer:

The GIMP application core, and other portions of the official GIMP distribution not explicitly licensed otherwise, are licensed under the GNU GENERAL PUBLIC LICENSE – see the 'COPYING' file in this directory for details.

La biblioteca libgimp se distribuye con licencia LGPL:

'libgimp' and the other GIMP libraries are licensed under the GNU LESSER GENERAL PUBLIC LICENSE – see the 'COPYING' file in the libgimp directory for details.

Y añaden esta nota para intentar aclarar qué es un trabajo derivado o combinado y cómo afecta la licencia a estos trabajos:

*[The below explicit exemption, we hope, clears up the GIMP developers' position concerning an ambiguity with the GNU General Public License concerning what constitutes a 'mere aggregation' versus a combined or derived work. The intention is to make it clear that arbitrarily-licensed programs such as **GIMP plug-ins do not automatically assume the GNU General Public License (GPL)** themselves simply because of their invocation of (or by) procedures implemented in GPL-licensed code, via libgimp or a similar interface to methods provided by the pdb:]*

** If you create a program which invokes (or provides) methods within (or for) the GPL GIMP application core through the medium of libgimp or another implementation of the 'procedural database' (pdb) serial protocol, then the GIMP developers' position is that this is a 'mere aggregation' of the program invoking the method and the program implementing the method as per section 2 of the GNU General Public License.*

GIMP usa TinyScheme que se distribuye con una licencia tipo BSD modificada, en la que tan sólo se exige que se mantenga el copyright y el disclaimer en los trabajos derivados y que no se use el nombre de los autores para promocionar estos trabajos derivados:

plug-ins/script-fu/tinyscheme – BSD-style-license
plug-ins/script-fu/ftx – BSD-style-license

He descargado la versión inestable actual, v 2.5.0 para ver si habían actualizado su licencia a la versión 3 GPL, pero no es así. Sigue distribuyéndose con la versión 2 GPL.

3. Modelos de negocio

Hasta la versión 0.99 Peter y Spencer se encargaron del desarrollo en su tiempo libre y no existía ninguna financiación.

Tras la marcha de Peter y Spencer, en el año 1998 se funda WilberWorks, Inc. Esta empresa, fundada por uno de los desarrolladores del proyecto, Zach Beane, se dedicaba a vender CD's de GIMP, en los que añadía trabajos artísticos y algunos programas y a trabajar como intermediario entre empresas que querían Plug-ins personalizados o nuevos desarrollos y los desarrolladores. También firmaban contratos de soporte, y prometían arreglar cualquier bug en 10 días para sus clientes.

De esta forma algunos desarrolladores y el personal de soporte obtenían pequeños ingresos por su trabajo en el proyecto. En palabras de Seth Burgess *"enough for a beer every now and then, but not enough to call it a living."*

He estado investigando pero no he encontrado apenas información sobre la desaparición de WilberWorks. Lo he intentado en su antigua web, en buscadores y en las listas de correo, pero no he encontrado casi nada. De hecho en la lista de desarrolladores alguien hizo una pregunta sobre WilberWorks, comentaba que estaba buscando información y que no encontraba nada, y no se le facilitó mucha información. Incluso un developer le contesta: *wilberwhat?*
<http://www.mail-archive.com/search?q=wilberworks&l=gimp-developer%40lists.xcf.berkeley.edu>

Actualmente el proyecto recauda dinero de donaciones (por cheque, transferencia bancaria o paypal), realizadas a través de la Fundación GNOME, una organización sin ánimo de lucro, exenta de impuestos, por lo que todas las donaciones realizadas son deducibles en la declaración.

Puede consultarse la lista de sponsors en la web del proyecto:
<http://www.gimp.org/donating/sponsors.html>

También reciben dinero de empresas que venden productos de merchandising de GIMP (CD's, pins, camisetas...) y donan parte de sus beneficios al proyecto.

En la web de GIMP comentan *"las ayudas financieras son necesarias por muchas razones. La convención anual de desarrolladores de GIMP es una de las cosas en las que se invierten las donaciones. Nos ayuda a traer a la conferencia al mayor número posible de desarrolladores."*

Un tema recurrente, que aparece a menudo en las listas y que se trata en casi todas las GIMP Developers Conference, es el de la creación de la Fundación GIMP, pero hasta el momento no se ha llegado a un acuerdo.

4. Motivación de los desarrolladores

Estudiando la lista de correo de los desarrolladores pueden formarse varios grupos en función del correo electrónico utilizado:

- algunos desarrolladores tienen mail de gtk.org y de gnome.org.
- muchos usan mail gratuitos: gmail, yahoo, hotmail.
- algunos desarrolladores usan correos de universidades (upv, informatik.uni-mannheim.de) y de centros de investigación (csail.mit.edu, boulder.swri.edu, am.ub.es)
- muchos usan el correo de su empresa: aol.com, inwise.de, restlesslemon.co.uk, ionflux.org, redhat.com, ximian.com, mmiworks.net, imendio.com
- algunos usan correos de sus web personales:mukund.org, ve3syb.ca
- y muchos otros tenían el correo de gimp.org.

El uso del correo corporativo en la lista indica conocimiento por parte de la empresa de que el desarrollador colabora en el proyecto, probablemente en horas de trabajo; por lo que es más que posible que la empresa en cuestión obtenga ciertos beneficios. En el caso de Red Hat o Ximian es evidente, ya que GIMP se incluye en sus distribuciones. Veremos el caso de otras empresas, como Imendio, más adelante.

Pero comenzaremos el estudio de las motivaciones con otros desarrolladores, aquellos que usan correos personales. Se muestran a continuación algunas frases o anécdotas, que representan diferentes motivos por los que los desarrolladores colaboran en este proyecto:

Kevin Cozens, <http://www.ve3syb.ca/me/index.html>

When not doing paid work I still spend a fair amount of time in front of a computer. I might be learning more about 3D modelling and ray-tracing, or working on bug fixes or modifications for some Open Source programs

Algunos de los proyectos con los que ha colaborado son: GEGL, TinyScheme, Morphix o WINE.

Parece que Kevin colabora con GIMP en su tiempo libre porque disfruta colaborando en este tipo de proyectos y porque aprende cosas nuevas con el desarrollo.

Mukund Sivaraman, <http://www.aboutus.org/MuKund.org>

Freedom of knowledge and expression (and as colloraries, free software and free will) are important for otherwise the world can't hold on to any of these. C. S. Lewis wrote, "Because free will, though it makes evil possible, is also the only thing that makes possible any love or goodness or joy worth having. A world of automata—of creatures that worked like machines—would hardly be worth creating."

Mukund parece que cree firmemente en el Software Libre, en las ideas de libertad que propugna y por ello disfruta colaborando en proyectos como éste.

Algunos de los desarrolladores de GIMP son españoles. Uno de ellos, Pedro Alonso Ferrer, comenzó a colaborar con GIMP en el Google Summer of Code del año 2006, con el proyecto "*Perspective Clone Tool for GIMP*" (estos proyectos están remunerados con 4500 USD). En ese momento estudiaba el 4º curso de Informática en la Universidad Pompeu i Fabra de Barcelona. Actualmente trabaja en la Universidad en el Departamento de Procesamiento de Imágenes. (<http://pedroalonso.net/blog/?p=25>)

Hay varios desarrolladores de GTK+ y de GNOME que colaboran también con GIMP, ya que los tres proyectos están muy relacionados. En la web de GTK se puede consultar dónde trabajan sus principales desarrolladores (<http://www.gtk.org/development.html>). Cinco de los 10 desarrolladores principales del proyecto trabajan en Novell y Red Hat, dos de los mayores financiadores.

Para terminar, investigué un poco sobre las dos personas que aparecen en la web de GIMP como maintainers, Michael Natterer y Seven Neumann.

Michael Natterer, que también es uno de los principales desarrolladores de GTK y colabora con GNOME, trabaja en la empresa Imendio. La verdad es que no me sonaba de nada esa empresa, así que me fui a la web (<http://imendio.com/>), y la primera frase de la página principal es la siguiente:

"Imendio is a small European company employing some of the world's most experienced and talented GTK+ developers..."

Se trata de una empresa que desarrolla aplicaciones software para sus clientes basadas en soluciones GLib/GTK+, y cuenta en su plantilla con algunos de los principales desarrolladores de estos proyectos.

Sven Neumann es el desarrollador más activo del proyecto. Comenzó a colaborar con GIMP cuando era estudiante en el año 1998. En el año 2000 escribió el libro *GIMP Pocket Reference*, que sigue editándose y a la venta. También es desarrollador de otros proyectos como GNOME, GTK o Pango entre otros.

Este apartado se quedaría incompleto si no hablásemos de las causas que llevaron a los autores originales a comenzar el proyecto. En una entrevista muy interesante que ambos concedieron un año después de abandonar el proyecto, comentan lo siguiente:

Peter Mattis: *You should understand that the Gimp and Gtk weren't written to fill holes in the software available under the GPL and LGPL. The Gimp was started because I wanted to make a webpage. Gtk was started because I was dissatisfied with Motif and wanted to see what it took to write a UI toolkit. These are purely selfish reasons. That is probably why the projects progressed so far and eventually succeeded. I personally find it much more difficult to work on something for extended periods of time for selfless reasons.*

Spencer Kimball:*From the first line of source code to the last, Gimp was always my "dues" paid to the free software movement. After years of using emacs/gcc/linux/etc., I really felt that I owed a debt to the community which had, to a large degree, shaped my computing development.*

5. Reglas de gobierno

Las reglas de gobierno del proyecto han ido variando a lo largo del tiempo, a la vez que cambiaban los miembros del equipo de desarrollo.

Al comienzo del proyecto Peter y Spencer tomaban todas las decisiones. Tan sólo recibían de la comunidad informes de errores que les ayudaban mejorar la aplicación. Cuando terminaron sus estudios en la universidad comenzaron a trabajar y abandonaron el proyecto.

Federico Mena Quintero (co-fundador junto a Miguel de Icaza de GNOME) se convirtió entonces, en 1997, en el mantenedor del proyecto. La primera decisión que tomó fue realizar una congelación de nuevas funcionalidades hasta que se consiguiera estabilidad.

Comenzó a formarse un nuevo equipo de desarrolladores que tenían funciones asignadas: *In the new system, there are designated team members; Manish Singh (yosh), for example, was in charge of making releases. Adrian Likins, maintaining data. Larry Ewing (lewing), Matthew Wilson (msw), and a host of others made bug fixes, and did other messier stuff. There's also a lot of overlap between the developer community of GIMP and other related projects. But they work as a team - nobody was, or is, the project leader. Each person makes their own contributions, and we all know who to refer to when we don't know how to do something, or want advice or options. Decisions that control the fate of GIMP are made primarily on **#gimp IRC channel**, through this team effort.*

No existía un líder del proyecto que tomara todas las decisiones, trabajaban en equipo y había un responsable para ciertas partes del proyecto, al que los demás se dirigían si tenían dudas o problemas. Aunque contaban con listas de correo para usuarios y para desarrolladores, las decisiones se tomaban discutiendo en el canal IRC #gimp. En una encuesta que Seth Burgess hizo entre los desarrolladores del proyecto cuando escribía '*A brief (and ancient) History of GIMP*' (de donde se ha tomado el párrafo anterior), se preguntaba si GIMP sería lo mismo de no haber existido el canal IRC; las respuestas variaban del 'Es probable que no' al 'No rotundo'.

A partir de 1999 otros desarrolladores comienzan a ganar más protagonismo, mientras antiguos desarrolladores van retirándose del proyecto y con la versión GIMP 1.2.0 en 2000, los desarrolladores más importantes son Manish Singh, Sven Neumann y Mitch Natterer.

A partir del año 2000 comienzan a celebrarse las GIMP Developers Conference, GIMPConf, en las que los desarrolladores se juntan unos días para conocerse en persona, discutir futuros desarrollos y conocer también a usuarios de GIMP. Aquí es donde se decide el roadmap, la lista de tareas principales, se habla de bugzilla, de las listas de correo, del sitio web, de la financiación, se decide cómo gestionar la distribución de plug-ins...

Sigue usándose el canal IRC y las listas de correo, y también bugzilla, para tomar decisiones, aunque las discusiones se hacen muy largas y tediosas, así que es difícil llegar a acuerdos. En la GIMPConf de 2003 se trata este tema:

Currently, there are two ways to get something done in the GIMP. First, you can write decent code and patches for a while, until you get CVS commit access, then you write whatever feature you're interested in, and commit it. If no-one backs it out, then it's in.

Second, you can bring it up on the mailing list, or in bugzilla, or in IRC (more on communication later), and discuss it until there is a consensus. However, we tend to be pretty bad at reaching consensus on anything even slightly contentious. The important questions to be answered any time a discussion like this comes up are what's going to be done, and who's going to do it.

It was agreed that beyond a certain point, there is generally very little technical merit to these types of discussions. It was felt that about 5 days was enough for everyone with an opinion on a given matter to make that opinion known, and that after that point, it would be an idea to have a summary of the salient points and close the discussion. That means, the people here would stop posting to the discussion. Of course, if other people want to keep on flaming, they're free to do so, but people will just ignore the thread.

At this point, the summary should sum up the various points, and finish with an answer to those two questions - what gets done, and who's doing it.

We may even have a bake-off system. If there are two competing ideas for the way something should be done, currently no-one tends to do it. Ideally, both people would do it and we pick the best one.

This brings up another point, though - in general, what is authority? And in particular, at what point has someone gained enough standing and authority to post one of these thread-ending summaries?

Cuando surgen discusiones por algún motivo, parece que les cuesta llegar a acuerdos y decidir qué hay que hacer y quién va a hacerlo. Tras esta conferencia se llega a la conclusión de que a los 5 días que se abre una discusión sobre un tema, todo el mundo ha podido opinar al respecto, así que a los 5 días se cerrará la discusión, con un resumen en el que se incluya las tareas a realizar y quién se encargará de ellas (regla "5 days and it's dead"). Pero les surge otra pregunta: ¿Quién tiene la autoridad para cerrar una discusión y realizar este resumen?

En esta conferencia se nombra un release manager, Dave Naery, y se definen sus funciones:

- *Follow CVS so that he can write release notes, and knows at any given time who is working on what, and at what stage it is*
- *Follow bugzilla closely*
- *Make releases regularly, according to the roadmap (or make sure releases get made)*
- *Update the roadmap to reflect reality*
- *Write release notes for the releases (keep NEWS up to date)*
- *Generally annoy people sending mails to the mailing lists and sending content to the website to explain the state of play and get people to work on stuff.*

En la última GIMPConf, celebrada en 2006, los maintainers del proyecto intentan hacer incapié en que los desarrolladores usen la lista de correo para tomar decisiones. Parece que la mayoría de las decisiones se toman discutiendo en el canal IRC o directamente poniéndose de acuerdo los implicados por mail personal u otros medios.

En resumen, las reglas de gobierno del proyecto han variado a lo largo del tiempo, a la vez que variaban los desarrolladores. Al principio todas las decisiones las tomaban Peter y Spencer.

Tras su marcha Federico Mena se convirtió en el mantenedor principal y se dedicó a intentar que GIMP ganara en estabilidad. Se formó un equipo de desarrollo con tareas específicas de manera que un desarrollador importante se encargaba de algún aspecto y el resto se dirigía a él cuando había dudas sobre ese tema. Las decisiones se tomaban en el canal de IRC.

A partir del año 2000 toman protagonismo otros desarrolladores que se encargan de mantener el proyecto, Manish Singh, Sven Neumann y Mitch Natterer. Surgen problemas a la hora de tomar decisiones, es difícil llegar a acuerdos. Se organizan reuniones anuales en las que los desarrolladores intentan llegar a acuerdos globales para todo el año, fijan el roadmap, hablan de financiación y de cambios importantes.

6. Historia de desarrollo

En esta sección vamos a estudiar el código fuente de GIMP y su repositorio y veremos cómo ha ido evolucionando en el tiempo.

6.1. Análisis del código fuente

Primero usaremos la herramienta sloccount, que cuenta las líneas de código del proyecto, y muestra estadísticas de los lenguajes de programación utilizados para su desarrollo, así como estimaciones del coste y esfuerzo necesario para llevarlo a cabo.

Los resultados obtenidos son los siguientes (para la versión estable actual gimp 2.4.5):

Número de líneas de código total:	640723
Años necesarios estimados para el desarrollo:	3,83
Coste total estimado necesario:	\$ 23,913,627
Estimación de número de desarrolladores:	46 aprox.

Resultados obtenidos en el análisis de sloccount

Puede sorprender el valor obtenido que indica los años estimados necesarios para el desarrollo, 3,83 años, que es ciertamente inferior a los más de 12 años de vida del proyecto.

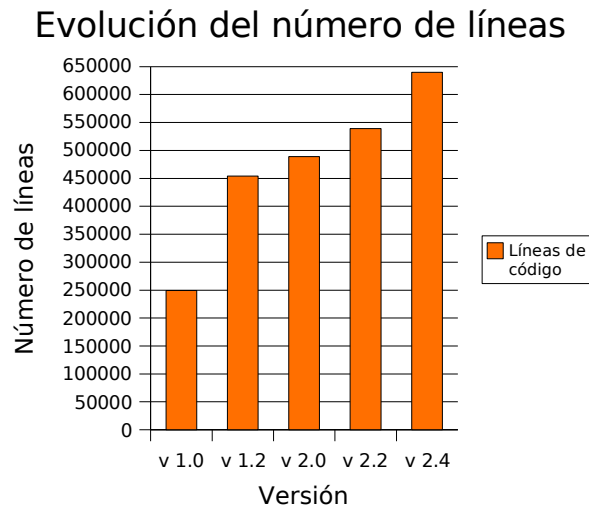
Este dato se explica si nos fijamos en el número medio de desarrolladores a tiempo completo, 46, necesarios para implementar el proyecto, que, como veremos en el próximo apartado, es muy superior al que dispone GIMP.

El lenguaje más usado con mucha diferencia es C, como puede verse en la siguiente tabla:

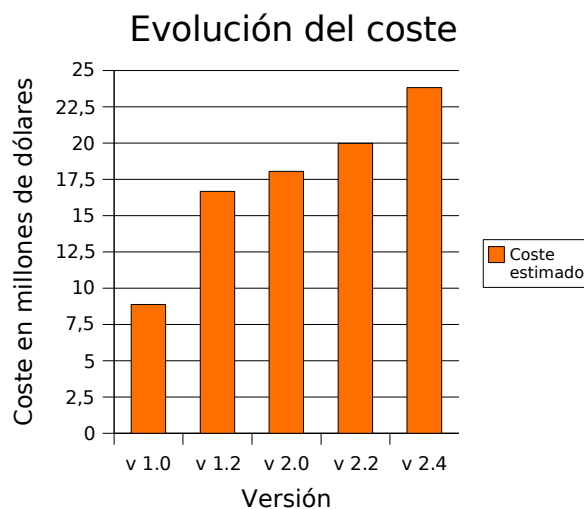
Lenguaje	Líneas	Porcentaje
ansic	621077	(96.93%)
lisp	11259	(1.76%)
python	3807	(0.59%)
perl	3266	(0.51%)
yacc	554	(0.09%)
sh	421	(0.07%)
lex	339	(0.05%)

Número de líneas de código agrupadas por lenguaje de programación

En las siguiente gráfica vemos cómo ha ido creciendo el número de líneas con las nuevas versiones lo largo del tiempo:



En la siguiente gráfica vemos cómo ha ido evolucionando el coste estimado de desarrollo, según los datos aportados por sloccount, que usa COCOMO básico para sus cálculos:



Para terminar con el análisis del código, nos fijamos en un detalle, el número de comentarios. En el análisis del proyecto realizado por ohloh.net, indican que el número de comentarios del proyecto es bastante bajo (un 13%) si se compara con otros proyectos escritos en C y C++ (la media en estos proyectos ronda el 20%). Esto puede indicar que el equipo de desarrollo no es muy disciplinado, lo que puede ser debido a la forma de gobierno y los procedimientos de desarrollo.

6.2. Análisis del repositorio

Otra forma de analizar el desarrollo de un proyecto es estudiar el repositorio del código. Para ello se ha utilizado la herramienta cvsanaly. Se muestran a continuación algunos resultados obtenidos al analizar la BBDD generada:

NOTA: En el repositorio sólo se encuentran los datos desde la versión 1.0. De las versiones anteriores no se disponen datos.

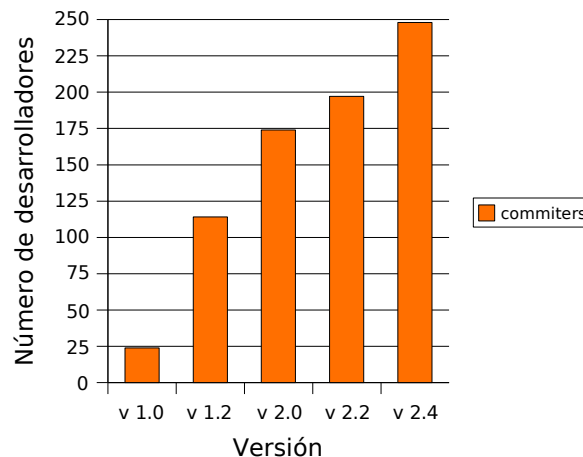
Número total de commits:	186,14
Número de módulos:	515
Número de desarrolladores:	248
Número de ficheros:	10,84
Primer commit:	24/11/97 23:05
Último commit:	28/05/08 16:33

Datos obtenidos por cvsanaly

Estos datos son globales, de (casi) toda la historia del proyecto. Indican que mucha gente ha participado en su desarrollo, 248 developers, a lo largo de estos 11 años.

En la siguiente gráfica vemos cómo ha ido evolucionando el número de desarrolladores que han colaborado con el proyecto a lo largo del tiempo:

Evolución de los desarrolladores



Se puede observar que tras la liberación de la primera versión estable, en la que colaboraron tan sólo 24 desarrolladores, se produjo un salto enorme, y se llegó a los 114 desarrolladores en la versión 1.2. Esta progresión se mantiene en el tiempo, hasta la versión 2.0., fechas en las que se estanca el número de personas que se acercan al proyecto para colaborar.

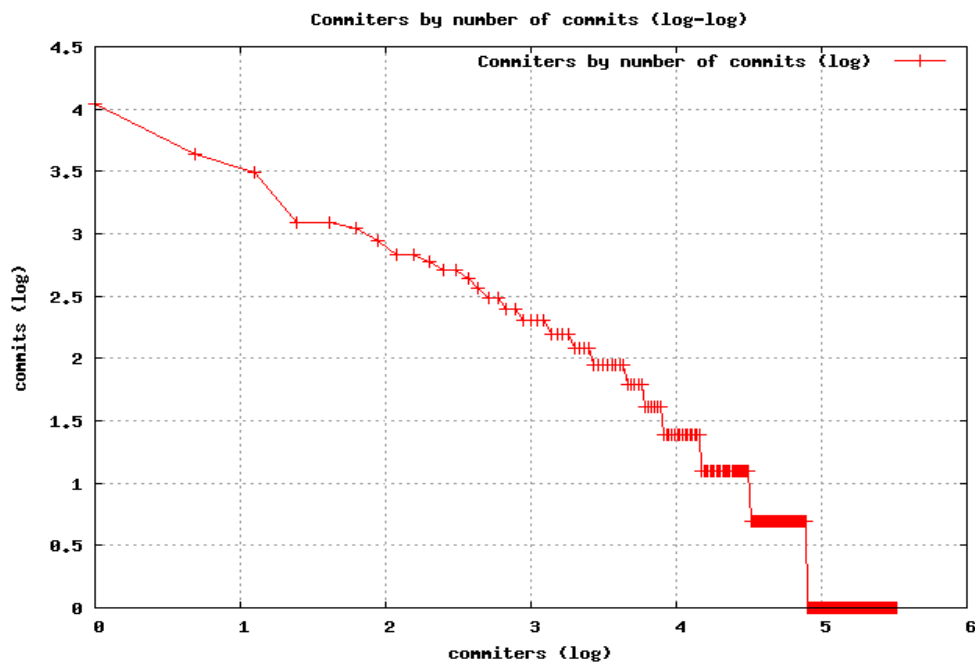
Con las últimas versiones se ha aumentado, de nuevo, la incorporación de nuevos desarrolladores, ya que se ha trabajado en este sentido, con manuales y FAQ preparados para la gente que quiere comenzar a colaborar con el proyecto. Se han creado guías de estilo y plantillas, y desde hace un año, se cuenta con especificaciones formales para la Interfaz de Usuario. Actualmente se está en

proceso de elaborar una lista de tareas sencillas para que los nuevos desarrolladores puedan comenzar a trabajar sin sentirse desbordados y sin tener que 'molestar' a desarrolladores más avanzados. Pero veamos cuántos desarrolladores han colaborado en los últimos 12 meses, lo que nos indicará la vitalidad del proyecto actualmente:

31 desarrolladores.

Según las cifras de ohloh.net, que analiza miles de proyectos, GIMP se encuentra en el top 2% de todos los proyectos en este sentido. Tiene, por tanto, un amplio equipo de desarrolladores que se mantienen activos.

En la siguiente gráfica puede observarse que la mayoría de los desarrolladores realizan muy pocos commits, y que son unos pocos los que se encargan de gran parte del trabajo.



Si sumamos los commits realizados por el top 10 de los desarrolladores de GIMP, obtenemos el 83% del total de commits del proyecto.

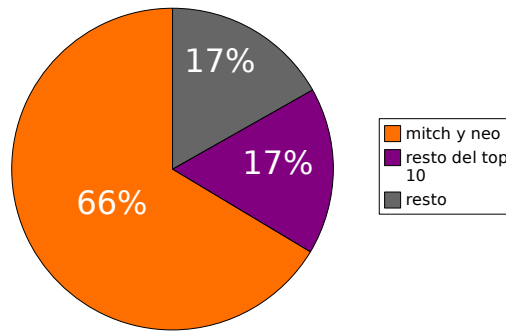
Desarrollador	Commits	Fecha incorporación
mitch	67929	Febrero de 1999
neo	55843	Febrero de 1998
yosh	15049	Diciembre de 1997
pcg	3750	Octubre de 1998
tml	2565	Enero de 1999
sopwith	2508	Noviembre de 1997
weskaggs	2047	Junio de 2004
simonc	1847	Junio de 2000
adrian	1588	Diciembre de 1997
dindinx	1545	Octubre de 2001

Top 10 de los desarrolladores de GIMP

De hecho, mitch y neo suman el 80% de los commits realizados por el top 10, es decir, que entre ellos dos han realizado el 66% del total de commits del proyecto.

La siguiente gráfica muestra los resultados comentados:

Porcentaje de commits



Por tanto Internet se ha convertido el canal básico para la comunicación de los miembros de la comunidad:

El sitio web oficial (www.gimp.org) está muy actualizado y dispone de mucha documentación y orientaciones para los usuarios noveles que quieren conocer GIMP o comenzar a colaborar con la comunidad. Se han desarrollado varios FAQ y HOWTO, en los que se enseña cómo informar de un error, cómo solucionar un fallo, cómo colaborar con la web... Dispone de enlaces a sitios elaborados por grupos de usuarios en distintos idiomas desde donde descargar manuales, obras, etc. Una de ellas es el sitio del grupo de usuarios de GIMP en español (<http://www.gimp.org.es/>) que tiene 1885,5 visitas diarias.

La comunidad dispone de varias listas de correo:

- GIMP User: para el usuario diario. Aquí podrás encontrar trucos para lograr efectos *cool*, preguntas de otros usuarios, etc.
- GIMP Announce: anuncios de las nuevas releases y de los nuevos plug-in.
- GIMPwin-users: para usuarios de la versión windows de GIMP
- GIMP Developer: para desarrolladores de plug-ins y del núcleo del programa; discusiones sobre el código fuente.
- GEGL Developer: para desarrolladores interesados en contribuir a GEGL, la nueva arquitectura para procesamiento de imágenes.
- GIMP web: para hablar sobre la estructura y el contenido del sitio web.
- GIMP docs: para hablar sobre el GIMP user manual y todo el material derivado.
- GIMP images: es un grupo creado para intercambiar imágenes creadas o manipuladas con GIMP
- GIMP-Perl
- GIMP-OS2

También disponen de varios canales de IRC. Uno de ellos, <irc://irc.gimp.org/#gimp>, es el canal principal de los desarrolladores y lleva funcionando desde 1997. Pero existen otros canales, para que los usuarios pregunten y resuelvan dudas, para discutir de la web ...

Otra de las herramientas de comunicación es el sistema de seguimiento de errores. Para ello se usa Bugzilla, <http://bugzilla.gnome.org/>. De hecho se usa la misma instancia para GNOME, GTK+ y GIMP. No sólo se usa para informar de los errores, sino que las sugerencias para nuevas funcionalidades también se escriben aquí.

La comunidad también cuenta con un Sistema de Control de Versiones, <http://svn.gnome.org/viewvc/gimp/>, que se encuentra en el servidor Subversion de GNOME.

A parte de la comunicación a través de Internet, desde el año 2000 se celebran reuniones en las que los desarrolladores y los usuarios se conocen en persona y charlan y discuten sobre el presente y futuro del proyecto. Hasta el momento se han celebrado 4 GIMP Developers Conference, y varios encuentros en el marco de otras conferencias como GUADEC o Libre Graphics Meeting.

8. Prácticas y procedimientos de desarrollo

GIMP es un proyecto un tanto anárquico en relación a los procedimientos de desarrollo. Hasta los últimos tiempos, en los que se está intentando mejorar en este sentido, no se fijaba un roadmap, y cada desarrollador decidía qué funcionalidades añadir, cuándo y cómo.

Las fechas de las liberaciones, por tanto, no están fijadas, y sobre la marcha se va decidiendo en la lista de desarrolladores cuándo se produce la liberación de la nueva versión.

Tampoco se fijan plazos fijos de congelación de nuevas funcionalidades, de cadenas traducibles o de la interfaz. Para cada versión, unas semanas antes se comienza a discutir en la lista y en el canal de IRC y se decide a partir de qué fecha comienza la congelación.

No han existido tampoco procesos de pruebas estructurados, ni he encontrado ningún desarrollador que se encargue en especial de este tema. Antes de la liberación de cada nueva versión se solicita en la web, en las listas y en los foros gente que quiera realizar pruebas.

En la web de los desarrolladores (<http://developer.gimp.org/>) se puede encontrar información y ayuda para los nuevos desarrolladores, con guías de estilo, recomendaciones, e incluso, plantillas para mantener la coherencia de estilo en el código.

En los últimos tiempos esta situación está cambiando algo. En la Conferencia anual se intenta fijar un roadmap, que marque el camino a seguir, y se está intentando obligarse a liberar una nueva versión cada 6 meses.

Desde el año 2006, GIMP se encuentra en un proceso de rediseño de la Interfaz de usuario para mejorar la usabilidad, y ha encargado a un equipo externo que se encarga de dirigir el proceso. Este equipo sí está llevando a cabo pruebas estructuradas, tests de usabilidad, etc. (Por cierto, estos tests de usabilidad han levantado un revuelo muy grande, debido a que la persona encargada de realizar los tests no es usuaria de GIMP, y esto ha molestado a algunos de los desarrolladores).

Se ha creado un wiki http://gui.gimp.org/index.php/GIMP_UI_Redesign y un sitio web para que los usuarios colaboren con ideas: <http://gimp-brainstorm.blogspot.com/>, pero las decisiones las toma este equipo, formado por 4 personas expertas en la construcción de interfaces persona-ordenador. Son ellos los que fijan el roadmap a seguir en la UI.

Desde la versión 2.4 han comenzado a trabajar fijando una especificación de requisitos para la Interfaz de Usuario, y parece que esta forma de trabajar ha dado buenos resultados. De hecho, se ha notado también en que ha aumentado el número de nuevos desarrolladores que se animan a colaborar con el proyecto.

9. Bibliografía

[1] www.gimp.org

[2] Seth Burgess. A brief (and ancient) history of The GIMP.

[3] www.ohloh.net/projects/gimp

[4] <http://en.wikipedia.org/wiki/GIMP>

Apéndice A

Anuncio de la publicación de la versión beta de The GIMP.

From: Peter Mattis
Subject: ANNOUNCE: The GIMP
Date: 1995-11-21
Message-ID: <[48s543\\$r7b@agate.berkeley.edu](mailto:48s543$r7b@agate.berkeley.edu)>
Newsgroups:
comp.os.linux.development.apps,comp.os.linux.misc,comp.windows.x.apps

The GIMP: the General Image Manipulation Program

The GIMP is designed to provide an intuitive graphical interface to a variety of image editing operations. Here is a list of the GIMP's major features:

Image viewing -----

- * Supports 8, 15, 16 and 24 bit color.
- * Ordered and Floyd-Steinberg dithering for 8 bit displays.
- * View images as rgb color, grayscale or indexed color.
- * Simultaneously edit multiple images.
- * Zoom and pan in real-time.
- * GIF, JPEG, PNG, TIFF and XPM support.

Image editing -----

- * Selection tools including rectangle, ellipse, free, fuzzy, bezier and intelligent.
- * Transformation tools including rotate, scale, shear and flip.
- * Painting tools including bucket, brush, airbrush, clone, convolve, blend and text.
- * Effects filters (such as blur, edge detect).
- * Channel & color operations (such as add, composite, decompose).
- * Plug-ins which allow for the easy addition of new file formats and new effect filters.
- * Multiple undo/redo.

Requirements -----

- * The operating system must support shared memory.
- * X11 R5 or R6. (Actually, it may work on R4, but we have not had a chance to test it).
- * The X-server must support the X shared memory extension. (The X-server does not actually need to support shared memory so this is only a temporary situation until we integrate the configure information with the source code).
- * Motif 1.2 or above.

The GIMP has been tested (and developed) on the following operating systems: Linux 1.2.13, Solaris 2.4, HP-UX 9.05, SGI IRIX.

Currently, the biggest restriction to running the GIMP is the Motif requirement. We will release a statically linked binary for several systems soon (including Linux).

URLs

<http://www.csua.berkeley.edu/~gimp>
<ftp://ftp.csua.berkeley.edu/pub/gimp>
<mailto:gimp@soda.csua.berkeley.edu>

Brought to you by

Spencer Kimball (spencer@soda.csua.berkeley.edu)
Peter Mattis (petm@soda.csua.berkeley.edu)

NOTE

This software is currently a beta release. This means that we haven't implemented all of the features we think are required for a full, unqualified release. There are undoubtedly bugs we haven't found yet just waiting to surface given the right conditions. If you run across one of these, please send mail to gimp@soda.csua.berkeley.edu with precise details on how it can be reliably reproduced.