UPGRADE
novática
Revista de la Asociación de Técnicos de Informática
NOVIEMBRE - DICIEMBRE 2001
154

**Novática, revista fundada en 1975, es el órgano oficial de expresión y formación continua de ATI (Asociación de Técnicos de Informática)**

ATI es miembro de CEPIS (*Council of European Professional Informatics Societies*) y tiene un acuerdo de colaboración con ACM (*Association for Computing Machinery*). Tiene asimismo acuerdos de vinculación o colaboración con AdaSpain, AI² y ASTIC

**<http://www.ati.es/novatica/>**



**Coordinación Editorial y Redacción Central (ATI Madrid)**
Padilla 66, 3º, dcha., 28006 Madrid
Tlf.914029391; fax.913093685 <*novatica@ati.es*>
**Composición, Edición y Redacción ATI Valencia**
Palomino 14, 2ª, 46003 Valencia
Tlf./fax 963918531 <*secreval@ati.es*>
**Administración, Subscripciones y Redacción ATI Cataluña**
Via Laietana 41, 1º, 1ª, 08003 Barcelona
Tlf.934125235; fax 934127713 <*secregen@ati.es*>
**Redacción ATI Andalucía**
Isaac Newton, s/n, Ed. Sadiel, Isla Cartuja 41092 Sevilla
Tlf./fax 954460779 <*secreand@ati.es*>
**Redacción ATI Aragón**
Lagasca 9, 3-B, 50006 Zaragoza
Tlf./fax 976235181 <*secreara@ati.es*>
**Redacción ATI Asturias-Cantabria** <*gp-astucant@ati.es*>
**Redacción ATI Castilla-La Mancha** <*gp-clmancha@ati.es*>
**Redacción ATI Galicia**
Recinto Ferial s/n, 36540 Silleda (Pontevedra)
Tlf.986581413; fax 986580162 <*secregal@ati.es*>

**Publicidad:** Padilla 66, 3º, dcha., 28006 Madrid
Tlf.914029391; fax.913093685 <*novatica.publicidad@ati.es*>

**Imprenta:** Gráficas Sierra S.L., Atenas, 3, int. bajos, 08006 Barcelona.
**Depósito Legal: B 15.154-1975**
**ISBN: 0211-2124; CODEN NOVAEC**

**Portada:** Antonio Crespo Foix / © ATI 2001

## SUMARIO



**Monografía del próximo número:** «***Gestión del Conocimiento y TIC***»

# La crisis del software libre científico

David Santo Orcero
*Consultor*

***<irbis@orcero.org>***

***Resumen:*** *el mundo científico produjo gran cantidad de software libre en los 70 y 80. Mas la producción de software libre se ha reducido en esta última década, hasta el punto de que hay áreas donde no se desarrollan herramientas libres desde los 80, o incluso donde no existen herramientas libres. En este artículo analizaremos las posibles causas y algunas propuestas de soluciones a este problema.*

***Palabras clave:*** *software libre, GPL, software científico*

## 1. La ciencia y el software libre

***Richard Stallman*** es la primera persona que ha definido el concepto de software libre. Después de la infructuosa relación con el software propietario que sucedió al abandono del **ITS** --libre-- por el laboratorio de inteligencia artificial del **MIT**, Richard Stallman dimitió de su posición y creó la **FSF** (*Free Software Foundation*) para difundir el software libre, tarea que comenzó con su célebre manifiesto (<http://www.gnu.org/gnu/manifesto.html>) y viene realizando desde entonces.

De cualquier modo, aunque Richard Stallman es el responsable de formalizar el concepto de software libre y de haber desarrollado la licencia **GPL** *(General Public Licence,* <http://www.gnu.org/copyleft/gpl.html>, de distinguirlo claramente del software gratuito y de haber sido uno de sus principales teóricos, no es el primer programador en desarrollar software libre. En el ámbito científico encontramos grandes proyectos colaborativos de desarrollo de software libre anteriores a Stallman, como es el caso de **GAMESS**. Además, en las épocas en que el peso del software propietario era más grande en el mundo «comercial», la mayor parte del software libre se concentraba en el área científica.

Tradicionalmente, la ciencia ha sido uno de los nichos de mercado más importantes del software libre. Permitiendo al científico una más rápida verificación de sus teorías aprovechando código de los otros, y asegurando la posibilidad de acceder a el código a bajo costo --ya que hay una amplia tradición histórica de que los científicos compartan código apenas a cambio de reconocimiento--, el esfuerzo de desarrollo compartido ha permitido acelerar el adelanto científico en muchas áreas, como por ejemplo en la química computacional con el proyecto **QCPE** (*Quantum Chemistry Program Exchange*, <ftp://qcpe6.chem.indiana.edu/>) o el **CCL** (*Computational Chemistry List*, <http://www.ccl.net/>); y al mismo tiempo ha permitido que pueda ser realizada ciencia también en lugares con bajo presupuesto.

El científico es conceptualmente afín a muchos de los conceptos del software libre: la formación de una meritocracia, donde eres más importante cuanto más has contribuido --código, en el caso del software libre; descubrimientos, en el caso del científico. Los científicos están acostumbrados a hacer públicos sus conocimientos publicando en revistas, donde son analizados entre sus semejantes; como es el caso de los programadores de software libre. Por ello, gran parte del código desarrollado por científicos tenían licencias que por los criterios actuales pueden ser consideradas libres.

Sin embargo, la situación ha comenzado a mudar. En el área comercial, el software libre se hace popular hasta el punto de comenzar a plantar cara al software propietario. Por otro lado, en el área científica cada vez son más frecuentes los paquetes propietarios, y vemos al software libre ahogarse y retroceder lentamente. El software científico bajo licencia GPL en muchas áreas actualmente es, desafortunadamente, prácticamente inexistente.

## 2. La causa de la crisis del software científico

Hay varias razones que pueden ser propuestas como causa de la crisis. No son mutuamente excluyentes; de facto, es posible que la verdadera causa sea una mezcla de todas ellas. Sin embargo, todas tienen en común que son causadas por el método de valorar los avances científicos y el mecanismo actual de financiación de la actividad científica.

La ciencia en los últimos años se viene haciendo más competitiva. Por razones que quedan fuera del ámbito de este trabajo --quizás desempleo, quizás más valorización de la carrera científica-- cada vez son más las personas que al terminar su graduación se deciden por cursar un curso de posgraduación y que prestan concurso para la carrera docente en alguna universidad. El hecho de aumentar la competencia ha hecho que la competición se haga más difícil. Y, de entre los distintos factores que intervienen en un concurso --prueba didáctica,

**Autor**

**David Santo Orcero** estudió Ingeniería Informática. Descubrió el software libre en 1994, cuando comenzó a usar Linux. Desde entonces ha estado trabajando con software libre. Ha sido becario de investigación durante los últimos cinco años, desarrollando aplicaciones para biofísica y física del estado sólido. Ha intentado mostrar las ventajas de usar y producir software libre donde ha trabajado, y ha publicado más de 150 artículos. Actualmente trabaja de consultor. Página personal: <http://www.orcero.org/irbis>

experiencia docente...-- o que intervienen en la obtención de una beca de investigación --experiencia investigadora--, aquel en que es más fácil marcar diferencia entre personas de la misma promoción es el número de artículos publicados en revistas indexadas.

Por otro lado, entre los grupos ya consolidados la situación es similar. Cada vez existen más grupos de investigación y la financiación no ha crecido proporcionalmente, por lo que la competencia es cada vez mayor. En muchos países, no llega a manos del investigador una perra gorda si no presenta un rendimiento en publicaciones, vía artículos publicados en revistas indexadas.

Este sistema «publica o perece» perjudica a los que desarrollan software libre, dado que no existen revistas indexadas específicas de software libre, por lo que es difícil conseguir una publicación de una nueva herramienta desarrollada en una revista indexada. El único método es publicar en una revista de informática teórica, lo que no siempre es posible. En la práctica, esto beneficia a los que trabajan en informática teórica, ya que les basta con la idea, el algoritmo en pseudolenguaje y, en el peor caso, una comparativa. Desarrollar una herramienta libre completa involucra también codificación, documentación de uso, verificación de código y, muchas veces, una interfaz gráfico. Por otro lado, si no implementa un algoritmo nuevo, no puede ser publicado; de nada vale decir: «mi programa es el primer y único programa libre que hace tal cosa».

Si alguien ya planteó el problema, aunque no haya desarrollado la aplicación, él publicó y la aplicación completamente desarrollada, funcional, no generará ninguna publicación. Queda publicar en revistas de otra área; por ejemplo, publicar en revistas de biofísica los programas libres de biofísica. Sin embargo, contamos con los mismos problemas que en las revistas de informática teórica, junto con los problemas de que tenemos que hacer un trabajo bien por encima de la media de los de la revista por estar fuera de área. Por si esto fuera poco *handicap*, además, muchas universidades y agencias de fomento de investigación ponderan negativamente, o incluso ni valoran aquellos artículos indexados realizados en revistas de fuera del área --en nuestro caso, informática.

Además, los mecanismos para avalar el rendimiento son ahora a muy corto plazo. Por ejemplo, la agencia de becas que mi última beca de investigación exigía un informe anual de rendimiento científico, con número de publicaciones en revistas indexadas. Si el rendimiento no es bueno, perdía la beca. Este hecho me obligaba a mantener dos líneas de trabajo, una a corto plazo para mantener la beca y otra a largo plazo, que era el objetivo real de la investigación. En la práctica, era preciso el doble de trabajo que el de cualquier otro investigador del departamento para mantener una evaluación de que mi trabajo era aceptable.

Por otro lado, los proyectos de software científico que deben ser encarados actualmente en muchas áreas --física, química, medicina, entre otras-- son de gran porte, necesitando años para su desarrollo.

Antiguamente era rentable para un grupo consolidado tener uno o varios científicos trabajando a tiempo completo desarrollando software libre. El placer de crear, de descubrir, suponía ya suficiente aliciente como para perder un poco de rentabilidad científica durante unos años, ya que parte del grupo estaba «entretenida» en tareas que no iban a suponer publicaciones --desarrollo de software científico. Sin embargo, actualmente, cada vez se da menos este caso. La supervivencia del grupo y el mantenimiento de la beca de cada uno de los investigadores depende de las publicaciones con vista a un año. Esto supone que en los grupos muchas veces se prioriza la investigación con resultados inmediatos. Rob Pike, en su artículo *Systems Software Research is Irrelevant* <http://www.cs.bell-labs.com/who/rob/utah2000.ps> sobre por qué la investigación de Informática de Sistemas estaba muerta, ya daba la voz de alarma. En áreas que no sean la informática, como es el caso de la ciencia base, la investigación en herramientas informáticas sufre más todavía de los problemas expuestos por Rob Pike.

La licencia GPL, aunque excelente, no ayuda tampoco a desarrollar software científico libre. Por un lado, es legal según la GPL cambiar el nombre del autor --aunque no sea ético. La GPL no permite realizar restricciones adicionales a la licencia, cuando sería necesario establecer algún mecanismo para que se citen los trabajos del autor del programa. Por otro lado, a diferencia de otras áreas, no tiene sentido en muchas áreas de ciencia base abrir una empresa de servicios de valor añadido. Aunque esto funcione bien en algunos casos --como es el caso de **HelixCode**, o de cualquiera de las empresas de distribuciones-- la masa crítica de posibles usuarios es lo suficientemente baja como para que no sea posible la supervivencia dando consultoría. Sí, existen universidades y algunas empresas con grandes paquetes de software cerrado, pero esos paquetes llevan años desarrollándose --algunos más de 20 años--, con el principio de que no tenían competencia; y cada día que pasa, los paquetes son mejores y es más difícil que el software libre pueda presentar competencia real.

Un ejemplo es el **Whatif**. Es un excelente paquete de ingeniería de proteínas, pero es propietario. Existen proyectos que intentan producir sustitutos, como es el caso de **Platon**, que son lo suficientemente poco conocidos y lo suficientemente poco desarrollados como para no ser tenidos en cuenta. Y cada vez la distancia se hace más grande, y ahora que el Whatif se ejecuta maravillosamente en **Linux**, es probable que acabe con la competencia. Esto supone que en un área tan crítica como el diseño de medicinas a partir de proteínas sintéticas no contaremos con software libre plenamente desarrollado. Por tanto, una primera causa para esta desaceleración del crecimiento del software científico es el sistema de valoración del rendimiento científico. Mas aunque seamos lo suficientemente idealistas como para jugar a la ruleta rusa con nuestra beca, las cosas no suelen ser tan simples. Universidades y centros de investigación suelen estar rígidamente jerarquizados. A diferencia de estructuras como el ejército, donde desde el principio está claro quien es el jefe, en los centros de investigación las relaciones de poder suelen ser delicadas, difusas y necesitan de una gran cantidad de diplomacia. Modificar la licencia de un proyecto científico para licencia

libre supone, como mínimo, convencer al orientador y al jefe de grupo de investigación de la idoneidad de liberar un código a cambio de nada. También hay que convencer a los compañeros de grupo, que ayudan incorporando su conocimiento a nuestra aplicación. En caso de responsabilidades compartidas --varios jefes--, la cosa se complica. En el momento que uno ponga reticencias, todos se cerrarán en banda. Y si el viejo gurú del laboratorio dice que eso no sirve para nada, necesitaremos un milagro para continuar con el proyecto.

## 3. Posibles soluciones al problema del software científico

La primera solución puede ser la existencia de una licencia libre, mas que recoja las singularidades del software científico. Esta licencia debería compartir el espíritu de la GPL y gran parte de su letra; y debería asegurar la preservación de las libertades propuestas por Stallman. Por otro lado, también debería incluir algunas restricciones adicionales, fáciles de cumplir y que no limitan la libertad final del código según los criterios expuestos por Stallman en «Qué es el Software Libre» <http://www.gnu.org/philosophy/free-sw.es.html>, como son:

· Si se usa el programa para realizar una investigación, y esa investigación termina en uno o varios artículos publicados, los artículos de dicha investigación deben reconocer qué programas fueron usados, así como citar los artículos en los que fueron descritos. Esta restricción no impide la libertad de libre uso, y, de hecho, la totalidad de los programas libres científicos la incluyen; este sistema hasta ahora parece funcionar. La principal objeción a esta restricción es similar a la crítica que Stallman hizo a la licencia BSD (disponible en <http://www.gnu.org/philosophy/bsd.html>); sin embargo, en este caso no se produce el exceso de verborrea en las citaciones; ya que, en caso de que alguien pueda publicar un artículo en una revista indexada, por la naturaleza de revisadas por asesores externos de estas revistas, ha de ser cierta importancia la aportación al código. Y para un cálculo en particular no es tan grande el número de citaciones derivadas. Por ejemplo, el **CCP4** es un proyecto libre, con modelo de desarrollo de bazar y una cantidad muy alta de personas que contribuyeron. Para un uso particular del programa --por ejemplo, cálculo de la estructura de una proteína por sustitución molecular-- es apenas preciso citar, a lo sumo, tres artículos. Observamos que cualquier uso no científico del software --como es el caso del uso educativo-- no entra dentro de esta restricción, y puede ser usado en condiciones iguales a la GPL.

· Si se modifica el código, se ha de identificar adecuadamente la parte modificada como tal. Esto no impide la libertad de modificación, mas permite que, en caso de que la modificación suponga una mejora en el rendimiento o la calidad de los resultados, el autor de la modificación tenga el debido reconocimiento; y, en caso de que la modificación introduzca errores, evita manchar el nombre del programa y del grupo que lo desarrolló. El hecho de comentar el cambio en el código fuente y en el *Changelog* es suficiente. Seguimos manteniendo el espíritu de la GPL, por la que la posibilidad de mejorar el código debe ser libre.

La segunda solución que puede tener el problema del software científico libre es la creación de una revista indexada destinada específicamente a productos cuya licencia sea libre. Debería ser todo lo amplia que fuera posible --es decir, no aceptando apenas la GPL, sino también licencias como la BSD, la artística o la de domínio público--, con la restricción de que la licencia del software asegure las libertades propuestas por Stallman, <http://www.gnu.org/philosophy/free-sw.html>. Sería una buena idea que dicha revista se hiciese ya con todos los requisitos para ser aceptada por indexadores e indexada, <http://www.isinet.com/isi/about/index.html>. Estos requisitos suelen ser de tipo lingüístico, desgraciadamente --el más común es que el título, las palabras clave y el resumen estén disponibles en inglés--. La mejor licencia para esta revista indexada de software libre sería la licencia GPL de documentación, <http://www.gnu.org/copyleft/fdl.html>. Esa hipotética revista de software libre publicaría artículos sobre software libre consistentes en descripciones de software libre, siendo como condición de publicación que el paquete esté terminado, funcional, y que los autores del artículo coincidan con los autores del código.

La tercera solución es el apoyo de las instituciones gubernamentales. En una primera solución, financiando proyectos libres, como ya hace el gobierno alemán, solo que en el área de software científico libre. Otra posibilidad es abrir grandes proyectos temáticos de software libre científico, como ya realizó el Reino Unido con los famosos **CCP**. Es fundamental el ofrecimiento de ofertas de becas sobre áreas informáticas de interés para los gobiernos, exigiendo que el código realizado lo sea bajo licencias libres.

La mejor solución, draconiana, es que aquellos proyectos financiados con dinero del estado deben ser libres. Esta restricción ya la tiene la legislación de los EEUU --y es por eso por lo que el GAMESS es libre--, y es de sentido común: si todos estamos pagando con nuestros impuestos el desarrollo de un programa, deberíamos beneficiarnos todos del desarrollo, y no apenas la institución donde se desarrolla el trabajo y el jefe del que desarrolla el código, que suelen ser los dos únicos beneficiados.

La última corresponde a la universidad: liberando los resultados de proyectos fin de carrera, de resultados de investigación y de tesis de doctorado bajo GPL de documentación, y el código desarrollado bajo una hipotética GPL científica con las características anteriormente comentadas, ganamos todos. La comunidad porque cuenta con más código libre para compartir y el investigador porque ve compensado su trabajo con el reconocimiento, los artículos y las citaciones que necesitará para conseguir becas y fondos para seguir investigando.