DRUG DISCOVERY TODAY BIOSILICO

Open-source software: not quite endsville

Matthew T. Stahl

Open-source software will never achieve ubiquity. There are environments in which it simply does not flourish. By its nature, open-source development requires free exchange of ideas, community involvement, and the efforts of talented and dedicated individuals. However, pressures can come from several sources that prevent this from happening. In addition, openness and complex licensing issues invite misuse and abuse. Care must be taken to avoid the pitfalls of open-source software.

Despite what Microsoft would have you believe, open-source software is a great thing. Open-source software powers many of the web sites on the Internet, corporate compute servers used for research and development, and a plethora of new gadgets that have broad appeal (beyond the most technically savvy members of the human race). Open-source software can be found in digital video recorders (Tivo), telephones, personal digital assistants (PDAs), watches, networking hardware, MP3 players and automobiles. Local and national governments are installing open-source operating systems and software on their computers, and sponsoring initiatives to foster the development of open-source projects [1–4]. Open-source software drives commerce, collaboration and communities. This article was written using an open-source word processor running on an open-source operating system. It is becoming a ubiquitous, if not always visible, presence in the world today.

The open-source development model responsible for such useful software largely succeeds on donated time and resources. It is remarkable that a worldwide community has been able to accomplish so much, and benefit society and commerce in so many ways. But FOSS (free and open-source software) is not a panacea for either users or developers. The barrier to creating a FOSS project is very low, which has predictably led to a proliferation of projects with a wide range of quality. Finding software that is useful, stable, well written and well documented amongst the ~87 000 projects hosted on SourceForge [5], a central FOSS repository, brings to mind the proverbial needle in a haystack. Even open-source heads of state Linus Torvalds [6] and Eric Raymond [7] call into question the utility of FOSS for the average user. Qualitative assessment of usability illustrates some of the issues [8,9]. Even for the more technically capable, unpolished open-source software can be difficult to use.

The ease of creating a FOSS project does not completely explain the proliferation. After all, many people have the opportunity to jump off a bridge yet choose not to. Developers are drawn to create FOSS projects by several powerful motivators [10]. Some projects are initiated to solve a specific problem, unsupported device drivers for example, and the results are shared with others. Creating or contributing to FOSS projects can provide an effective way to showcase a developer's talents. Employment offers and raises may come as a direct result of work on open-source projects. It can also be a hobby or creative outlet. A technically challenging or interesting problem may provide a level of mental stimulation that a software engineer's day job doesn't offer (the coding he or she does to relax after a long day of coding). Regardless of the original impetus for creation, the reality of managing a successful

Matthew T. Stahl
OpenEye Scientific Software,
3600 Cerrillos Road,
Suite 1107,
Santa Fe,
NM 87507, USA
e-mail: mstahl@eyesopen.com

1359-6446/04/$ – see front matter PII: S1359-6446(04)03364-1

FOSS project is quite daunting, and requires a great deal of effort and dedication. Professional quality is difficult to achieve with a 'hobby' level of effort. For other projects, the most difficult challenges are not technical, but legal, social and managerial. Inexperienced open-source software developers can easily be blindsided by a host of issues for which they are not prepared. Certain issues are common among all FOSS projects. Other challenges may be domain specific. Chemistry software exemplifies some of the common issues while presenting others that are unique.

Definitions

It is important, for the purposes of discussion, to define FOSS and open-source development. The Open-Source Initiative has an extensive ten-point definition of open-source software [11]. The Free Software Foundation (FSF) also defines the closely related 'Free Software' [12]. Free Software in this context is analogous to 'free speech' as opposed to 'free beer'. Whereas Free Software is more concerned with political and social aspects of software distribution than open-source software, both have two important traits: software is released in source-code form and is freely redistributable. Open-source software and Free Software will be considered synonymous. Open-source software is released under a license that defines the terms and conditions of use. FOSS is typically developed by communal effort, whereby people donate time and resources toward a common goal. This is called the open-source development model.

Talent pool

One of the common misconceptions when creating a new FOSS project is that it will be easy to attract the help of other developers. Successful open-source projects have hundreds, and sometimes thousands, of contributors. In reality, most projects comprise only a few regular contributors and quite often only a single developer [13,14]. This is especially true if the talent pool of developers is restricted to those with a unique area of specialization. Although knowledge of chemistry is not an absolute requirement for authoring chemistry software, it is difficult to attract developers willing to donate their talent unless they have an interest in the field. Conversely, molecular visualization has general appeal to software engineers interested in graphics; FOSS molecular viewers abound [15]. Electronic structure and molecular dynamics require a more specialized background, and therefore have a more difficult time attracting contributors. In the particular case of electronic structure calculations, there is an additional disincentive to open-source involvement as it can mean losing access to related commercial software [16]. Chemical informatics ('cheminformatics') is an area that might be expected to draw from the pool of talented developers employed by pharmaceutical companies but, in practice, the reclusive nature of the pharmaceutical industry discourages involvement in FOSS.

Intellectual property (IP) is the cornerstone of the pharmaceutical industry and they protect it tenaciously. Literally billions of dollars can be ascribed to holding patents and trade secrets. Given this high valuation of IP, it is natural that aggressive measures are taken to protect it. Patent infringement lawsuits are not the only mechanism used. Employees of pharmaceutical companies are asked to sign non-disclosure agreements at the time of their employment *de rigueur*. Exactly what constitutes IP that cannot be disclosed varies widely across the industry, but agreements often cover activities outside the workplace. This may include contributions to open-source software development projects within an employee's particular field, even if the work is not done with company time or resources. Significant contributions typically need to be reviewed by a corporate lawyer if there is any question of IP ownership. The requirement that potential contributions be approved by legal and research councils is a powerful deterrent. Even in cases in which contributions would clearly not constitute a conflict of interest, the barrier simply exceeds the would-be contributor's initiative.

This is not to say that employees of pharmaceutical companies do not contribute to open-source projects. In fact, they regularly contribute to projects that are relevant to their field. Relatively minor contributions, such as bug fixes, do not normally require oversight. Feature additions that do not constitute a significant development effort will not necessarily need explicit approval. But potential contributions viewed to be more than trivially valuable are subject to review and approval. This dynamic was clearly seen in the OELib project. OELib is a programming library used in the development of cheminformatics and modeling applications [17]. Employees of pharmaceutical companies made generous contributions to the project; however, there were numerous instances when significant improvements were not donated back to the main code repository. The general public license (GPL) under which OELib was released does not require this, and so these developers and their employers were well within their rights to withhold improvements. In doing so, however, the project was hurt in two ways. First, the improvements were not made available for public use, which would have benefited and encouraged the OELib community of users and developers. Second, the proprietary modifications caused that version of OELib to be cut off from the main development tree. It was typically too difficult to reconcile private versions with the 'official' OELib project, so people working with the proprietary versions became isolated from the rest of the community. Thus, even their smallest contributions and feedback were denied to the main project. In general, IP concerns can be detrimental to open-source development. This was certainly the case with OELib.

Distribution license

Selecting an open-source license is a minefield for which few are prepared when they need to be. There are a

plethora of licenses under which open-source software can be released [18]. Selecting a license at the initiation of a FOSS project is likely to be a low priority, as there is no initial value to the project. Without a line of source code written, wading through the legalese and nuances of distribution licenses seems unimportant. In reality, the irrevocable nature of the license makes this the most critical time if authors wish to eventually exercise control over derivative works. If no restrictions are to be placed on redistribution and derived works, then the software can be released into the public domain. This is rarely the case – hence the proliferation of open-source licenses. Unfortunately, even the most carefully selected and restrictive license may not afford complete protection from unanticipated and undesired uses.

Many open-source licenses, although not all, prevent the inclusion of open-source works in commercial software products. This is accomplished by requiring that derivative works be open source and freely redistributable. Care must be taken to choose an appropriate license if authors do not wish their work to advance the development of a commercial product. But the license itself does not guarantee protection from violation. The burden of enforcing the license against violators falls upon the copyright holders and the cost of enforcing a license would, in most cases, exceed the resources most FOSS projects have at their disposal. The FSF, from which the GNU GPL originates, offers legal assistance in cases of GPL violations when the copyright is assigned to the FSF. This assignment is a price that many open-source contributors do not wish to pay. Commercial software companies appropriating open-source software is not just a theoretical issue. There have been clear violations of the GPL [19–21]. Chemistry software is not immune. MDL based a product called 'Chime' on Roger Sayle's molecular visualization program 'Rasmol' without requesting permission from Sayle or his employer, GlaxoWellcome. Executives at GlaxoWellcome were not pleased with MDL [22]. According to Sayle, every other commercial use of Rasmol was preceded by at least an informal request. Although the first Rasmol release (1992) came after the initial draft of the GPL (1989), the GPL did not become widely known and adopted until years later, with the increasing popularity of Linux around 1995. A future release of Rasmol (version 2.8) will be licensed under the GPL.

Open-source licenses provide significant protection, as most commercial entities would not knowingly engage in license violation. Taking advantage of open-source software without violating the license can be trivial in some instances. Redistributing a stand-alone application based on open-source software as a 'helper' to the commercial tool is well within the rights granted by open-source licenses. The molecule file format interconversion program 'Babel' was redistributed by commercial software companies as a convenience to their customers. In so doing, the companies avoided having to modify their own programs to provide support for additional molecule file formats. This type of action is not a violation of typical open-source license agreements.

In chemistry and cheminformatics, data have value. Some companies exist just to generate, abstract and format chemical data. Spectra libraries, purchasable compounds and reaction databases have value, and are sold by several vendors. Open-source licenses do not restrict commercial end-uses of the software. If a database vendor wishes to use a freely available code to generate data for inclusion in a commercial product, they are free to do so. OELib was used in exactly this manner. It is difficult for authors of FOSS projects to anticipate potential end-uses, and even more challenging to prohibit undesirable uses unless developers are willing to author and enforce their own license agreement. Imposing restrictions and controls on the use of a FOSS project runs counter to the spirit of most licenses. It also discourages people from using the software, which defeats the purpose of an open-source license. Developers that are concerned with commercial end-uses should consider licenses other than open source.

Bioinformatics: a different landscape

It is generally perceived that open-source bioinformatics software has been more widely adopted and more successful than its counterparts in chemistry and cheminformatics. A stunning amount of freely available bioinformatics software and services are available on the Internet. Several articles have been published that discuss the impact of open-source software on bioinformatics [23–25]. In the course of preparing this article, I engaged in numerous conversations and debates on the topic, and yet the exact reasons for this success remain unclear. There may not be a single reason, but rather a combination of factors that have led to the wealth of useful open-source bioinformatics software. One distinct difference between cheminformatics and bioinformatics is the availability of data. A vast amount of genomic data are freely available on the Internet. By contrast, public domain chemical, or at least structural, data are scarce. The availability of bioinformatics data could be driving the development of open-source tools. Almost all successful open-source software is driven by a clear understanding of the specifications. Developers know the problem they are trying to solve and roughly how it should be solved. Systems software (kernels, compilers, web servers) directly benefit from clear specifications and a group of developers working toward a common goal. Bioinformatics software also benefits from such clarity. The data are contained in one of a small number of well-known formats, and it can be searched and analyzed in several well-defined ways. Chemistry, by contrast, is extraordinarily diverse. Getting computational chemists to agree on a single data representation, even when they are trying to accomplish similar goals, is difficult. There exists little hope of agreement among spectroscopists, quantum chemists, modelers and cheminformaticians on

any subject related to software development. With the lack of community and common goals, it is difficult to build up a critical mass behind any open-source development project. The commonality of goals in bioinformatics makes communal development feasible.

An economist would attribute the differences between the two fields to the result of market forces. Chemistry software has a long history of commercialization. Academics, performing research with the aid of federal research grants, routinely funnel technology and software into commercial enterprises. The chemical software market seems to bear endless new entries into the field, encouraging entrepreneurs. As there are few open-source alternatives to proprietary chemical software, this trend will probably continue. Bioinformatics is an increasingly difficult market for entry by commercial software. As open-source bioinformatics software proliferates, the cost of developing a system with enough commercial value to compete with non-commercial systems becomes prohibitive. Any successful commercial software will probably be met with direct competition from open-source projects. In chemistry, commercial software breeds commercial software. In bioinformatics, open-source software breeds open-source software. The origins of this trend in chemistry software can be rationalized by commercialization events in the late 1970s and early 1980s. By contrast, the National Center for Biotechnology Information (NCBI) was formed in 1988 at the cusp of explosive growth in bioinformatics. The creation of the World Wide Web protocols in 1991 provided the conduit for disseminating and searching sequence databases. Commercial bioinformatics software competes with the wealth of information and search tools available at the NCBI, the European Molecular Biology Laboratory (EMBL), the European Bioinformatics Institute (EBI) and the DNA DataBank of Japan (DDBJ). The existence of national and international genome research centers encourages open-source software development by undercutting commercial software.

Conclusions

Open-source software is still a relatively new phenomenon. People are in the process of learning how it will affect them personally and professionally. Only now are we beginning to see the social and commercial interplay of open-source software in chemistry and bioinformatics. It can be a very positive thing, but the choice to be involved in open source should be made with care. Unfortunately for neophytes, the body of knowledge on which to base their decision is still forming. This is especially true in specialized areas such as chemistry, for which few open-source projects have been attempted. General wisdom regarding open-source software is partially applicable to chemistry; however, some of the challenges and obstacles presented by chemical software are specific to the field.

The examples described will hopefully provide a realistic perspective on the issues involved in open-source software development in chemistry.

Acknowledgements

I would like to thank Anthony Nicholls, Roger Sayle and George Vacek for assisting in preparing this article.

References

1 Massachusetts state: Enterprise open standards policy on World Wide Web URL http://www.mass.gov/itd/openstandards.htm
2 Government Information Officers' Council: Using open-source software in the South African government (2003) on World Wide Web URL http://www.oss.gov.za/modules.php?op=modload&name=Downloads&file=index&req=getit&lid=6
3 Varghese S: ACT set to adopt open-source bill (2003) on World Wide Web URL http://www.theage.com.au/articles/2003/12/10/1070732274118.html
4 McMillan R: IBM, Brazilian government launch Linux effort (2003) on World Wide Web URL http://www.infoworld.com/article/03/10/10/HNibmbrazil_1.html
5 SourceForge.net on World Wide Web URL http://www.sourceforge.net
6 Mackenzie K: Linus Torvalds Q & A (2004) on World Wide Web URL http://www.google.com/search?q=cache:F3Ab57EcidAJ:www.australianit.news.com
7 Raymond ES: The luxury of ignorance: an open-source horror story (2004) on World Wide Web URL http://www.catb.org/~esr/writings/cups-horror.html
8 Smith S *et al.*: GNOME usability study report (2001) on World Wide Web URL http://developer.gnome.org/projects/gup/ut1_report/report_main.html
9 Eklund S *et al.*: StarOffice Calc versus MS Excel: improving the usability of an open-source spreadsheet application (2001) on World Wide Web URL http://www.sims.berkeley.edu/academics/courses/is271/f01/projects/StarCalc/
10 Hughes J: Why do people write open-source software? on World Wide Web URL http://www.openxtra.com/articles/why-do-people-write-open-source-software.htm
11 Perens B: The Open Source Definition (2004) on World Wide Web URL http://www.opensource.org/docs/definition.php
12 Free Software Foundation: The Free Software Definition (2004) on World Wide Web URL http://www.fsf.org/philosophy/free-sw.html
13 Zhao, L. and Elbaum, S. (2003) Quality assurance under the open source development model. *J. Syst. Softw.* 66, 65–75
14 Chromatic: Myths open source developers tell ourselves (2003) on World Wide Web URL http://www.onlamp.com/pub/a/onlamp/2003/12/11/myths.html
15 The OpenScience Project on World Wide Web URL http://www.openscience.org/index.php?section=56
16 Banned by Gaussian on World Wide Web URL http://www.bannedbygaussian.org
17 The Open Babel Project on World Wide Web URL http://openbabel.sourceforge.net
18 Open Source Initiative: The approved licenses (2004) on World Wide Web URL http://www.opensource.org/licenses/
19 Is Linksys shirking the GPL? (Maybe not) (2003) on World Wide Web URL http://www.oreillynet.com/cs/user/view/cs_msg/22344
20 Netfilter project was granted a preliminary injunction against Sitecom GmbH on World Wide Web URL http://www.netfilter.org/news/2004-04-15-sitecom-gpl.html
21 mp3machine.com on World Wide Web URL http://www.mp3machine.com/software/MP3Workshop/wwwboard/messages/1.shtml
22 Hodgson, J. (1996) GlaxoWellcome and MDL become entangled in the Web. *Nat. Biotechnol.* 14, 690
23 Stein, L. (2002) Creating a bioinformatics nation. *Nature* 417, 119–120
24 Bioinformatics World: Open doors for open source on World Wide Web URL http://www.lifescienceit.com/biwaut02secrecy.html
25 Freely available software plays special role for big pharma and others (2002) on World Wide Web URL http://www-106.ibm.com/developerworks/linux/library/l-osbio.html