

## THE DANGERS OF OPEN-SOURCE CODE

William H. Venema  
[Bill@TheVenemaReport.com](mailto:Bill@TheVenemaReport.com)

In his highly acclaimed best seller, *The World is Flat*, Thomas Friedman hails open-sourcing as one of ten “flatteners” of the world. But has the pervasive use of open-source software created ticking time bombs for the enterprises that have knowingly (and unknowingly) included open-source code in their proprietary software?

Open-source software is software for which the source code is freely and publicly available. Although open-source software is also called “free software” (see, [www.fsf.org](http://www.fsf.org) and [www.opensource.org](http://www.opensource.org)), it is far from free. There are specific licensing agreements that are applicable to such code, which specify the responsibilities of the user of the code.

### A COMPLEX SET OF RULES

It is not an easy task to determine the restrictions that open-source licenses impose on the use of particular open-source code. As of March 2008, the Open Source Initiative (“OSI”) had approved 68 different open-source licenses. To complicate matters further, each of the 68 licenses is different from the others. In addition to the licenses approved by OSI, there are hundreds of other versions of software licenses, from the idiosyncratic licenses created by particular developers to variations on the licenses approved by OSI. Finally, if developers combine open-source code from several sources, along with their

proprietary code, then they create a complex mix of intellectual property that is virtually impossible to decipher and may include conflicts that prevent its distribution at all. For example, software components licensed under the widely-used Gnu General Public License (“GPL”) may not be distributed with software components licensed under the Mozilla Public License.

Despite the numerous open-source licenses in existence, there are some that dominate the field. OpenLogic, an organization that provides enterprises with a certified library of open-source software, reported on the licenses used by the top twenty packages in its library.

<b>Licenses of the Top 20 Packages in the OpenLogic Certified Library *</b>	
GPL or LGPL	20%
Apache License	75%
CPL, Perl, Eclipse, and BSD Licenses	20%

\* Note that these percentages add up to more than 100% because several packages have multiple licenses.

## **THE DANGERS WITHIN**

The reason that open-source code has become a ticking time bomb for many enterprises is that any violation of the applicable open-source license is a violation of copyright law, which can give rise to liability for significant damages. The problem is especially acute when open-source code is incorporated into proprietary software, because a common goal of many of the open-source licenses is to preserve “openness” by requiring that any modified software be redistributed on the same terms as the open-source software and at no charge. Thus, if a programmer includes, in the proprietary software of

an enterprise, any open-source code that requires the modified software to be redistributed at no charge, then any subsequent distribution of the proprietary software must be free. Consequently, the commercial value of that proprietary software is thereby virtually eliminated.

The adverse effects of failing to comply with the applicable open-source licenses usually occur automatically. Several of the most frequently used open-source licenses are “conditional licenses,” which means they do not include standard notice-of-breach and cure provisions. Therefore, if a user fails to comply with the terms of the license, the license terminates automatically. Because the open-source code is protected by copyright, the license is the only source of rights to use the software. If the license terminates, then any continued use of the software constitutes copyright infringement. Public companies, in particular, cannot ignore such infringement and hope they avoid detection, because Sarbanes-Oxley and Statement 142 of the Financial Accounting and Standards Board (“FASB”) require public companies to value their software and assess their litigation risks.

The adverse effects of failing to manage the use of open-source code can extend beyond liability for damages for copyright infringement. For example, when IBM acquired Think Dynamics a few years ago, an examination of the software code of Think Dynamics found 80 to 100 examples of open-source code that the company had failed to account for properly. As a result, IBM reduced the purchase price for the company over thirty percent (30%)—from \$67 million to \$46 million. Discovering unaccounted for open-source code in an enterprise’s software during the due diligence process could also upset a potential joint venture, an investment in the enterprise, or other strategic transaction.

Accordingly, management practices must guard against not only potential violations of the copyright law (which can lead to liability for damages), but also the risk that a strategic transaction might be derailed during due diligence, because of the discovery of unaccounted for open-source code in the enterprise's software or products.

## **CHALLENGES TO MANAGING THE RISKS**

Based on the foregoing discussion, it is clear that enterprises should carefully manage the use of open-source code within their organizations. But if one is tempted to deal with the risks posed by open-source code by adopting a policy prohibiting its use, forget it. Attempting to do so would be futile and counterproductive. Enterprises are using open-source code at an accelerating rate. OpenLogic reports that in 2006, enterprises on average used 75 different open-source packages. That number grew to 94 in 2007.

One of the causes of this growth is the widespread adoption of the "software assembly" model to create software products, whereby developers create a product by using existing code from a variety of internal and external sources, thereby making it difficult to determine the ownership of the product. Another cause of growth is the use of offshore programmers, who like to use open-source code and mix it with the proprietary code they create for many American corporations. Although the software assembly model and the use of offshore programmers are welcomed in many quarters, because they permit software to be developed rapidly, thereby reducing development time and costs, such savings must be weighed against the risks associated with the uncontrolled use of open-source software.

## **DEVELOPING AND IMPLEMENTING A WORKABLE POLICY**

The use of open-source software brings with it risks that enterprises cannot ignore. And yet, enterprises want to use it. Accordingly, a good policy must balance the desire to use open-source code against the risks that it poses.

Regulating the use of open-source code could take a variety of forms. An open-source policy could range from a simple one for a company with relatively little need to use open-source software, to an elaborate structure that meets the needs of a public company. Typically, such policies are created and implemented by a committee that includes technical, business development, and legal representatives.

Such committees should have the responsibility and authority to govern all use of open-source code within the enterprise, including both internal use and the incorporation of open-source code into commercial products. Although such committees are usually tasked with reviewing and deciding requests to use open-source code, a committee can reduce the number of requests that it must review by categorizing certain items of open-source code. The committee could prepare a list of open-source code that, because of the licenses to which it is subject, may be used at any time and some code that may be used for certain purposes (*i.e.*, an “approved list”). On the flip side, the committee could also prepare a list of open-source code that, because of the licenses to which it is subject, may not be used at any time or for any purpose (*i.e.*, a “blacklist”). Open-source code that is on neither list would be subject to a review process to determine whether the enterprise should use the code at all and, if so, under what circumstances.

Although preparing an approved list and a blacklist will solve many of the problems associated with using open-source code, it is only a start. In making its decisions, the

committee should consider not only the provisions of the license that govern the use of the open-source code, but also how the open-source code is proposed to be used in the enterprise. Consequently, the committee must thoroughly understand the needs of the enterprise. Some uses of open-source code will not put the enterprise at risk, while other uses, such as incorporating it into commercially distributed software, could pose serious problems. The committee should also address the use of open-source code by outside contractors, including how the enterprise will monitor such use, as well as the contractors' compliance with the policies of the enterprise. Periodically, the committee should oversee an audit of some or all of the enterprise's software to determine what open-source code is present and whether the committee's management of the use of open-source code has been effective. Commercial products such as Black Duck and Palmida are available to assist with such audits.

## **CONCLUSION**

Clearly, open-source code holds great promise. It can help enterprises run their organizations more efficiently and bring their products to market more quickly and at lower cost. With millions of contributors around the world, many open-source packages offer products that are vastly superior to commercially available products. Nevertheless, such code also poses risks. In order to avoid the "time bombs" described above, enterprise executives must understand the provisions of the licenses that govern the open-source code being used within their organizations and develop effective policies for managing such use.