

Open Source Software Issues in Acquisitions and Other Inbound Transactions

Stephen Mutkoski

Intellectual Property & Licensing Group

Microsoft Corporation

Complex Open Source Software (OSS) issues can arise in the context of any inbound technology transaction, including acquisitions of entire companies, asset acquisitions and work for hire transactions. The presence of OSS in such transactions may raise significant copyright infringement issues and impact the purchaser's valuations of code, intellectual property assets and employees that the target entity would transfer. This paper outlines the issues and associated risks that can arise in a transaction where the purchaser learns of the presence of OSS (or other third party copyrighted code) in the target's assets. After introducing these issues, the paper offers some suggestions for streamlining the mergers and acquisitions diligence process to more effectively identify and resolve potential OSS issues early in the deal process.

I. Assessing Issues and Potential Risks

There are several distinct OSS related issues that should be considered in any inbound transaction involving the transfer of software by a third party.¹ The most important issue relates to the broad intellectual property grants included in so-called "copyleft" or "reciprocal" OSS licenses and issues that might arise if the purchaser were to distribute code governed by such a license. Another important issue relates to general concerns about the origin or "pedigree" of OSS code (whether released under a copyleft license or a more benign BSD-style license) and whether the purchaser is willing to take on these risks and continue shipping the code at issue.²

Copyleft or Reciprocal License Issues

Depending on the purchaser's business and licensing model, there may be significant issues relating to so-called "copyleft" or "reciprocal" licensed code that is included within the target's code base. These types of licenses require, as a condition of the license grant, developers who incorporate and distribute code covered by such licenses to grant: (1) broad access to code (in source code form); and (2) broad intellectual property licenses to all or part of the resulting work that the developer makes with the copyleft licensed software, allowing recipients to make and distribute unlimited copies.

When people say that copyleft licenses are "viral" they are referring (in a somewhat exaggerated fashion) to the requirement that copyleft licenses "impart" the copyleft license terms to other code that is combined with the copyleft code. The term "viral"

¹ Purchasers should also be on the lookout for other restrictive conditions in a source code license (whether the license qualifies as "open source" or not) that place limits on use or distribution (examples include "non-commercial use only" or "research use only") and accordingly might be inconsistent with the intended business and licensing model for the code.

² In addition to copyleft and pedigree issues, purchasers may also want to consider so-called "patent peace" clauses in certain OSS licenses. These clauses attempt to discourage assertion of patents, by terminating patent rights granted under the OSS license in the event a licensee asserts certain categories of patents. Depending on the size and breadth of their patent portfolio, purchasers may want to avoid using code that is licensed contingent upon not asserting some or all of their patents.

may be a bit of an overstatement because the copyleft code does not “infect” or “taint” other code merely by being combined with such code. Instead, if such a combination is made, the copyleft license requires the developer to license his other code under the copyleft license terms as a condition to using the copyleft code. If the developer fails to license his own code under copyleft terms, the developer has no right to use the copyleft code and could be sued for copyright infringement. The General Public License (GPL), Lesser General Public License (LGPL), Mozilla Public License (MPL) and Common Public License (CPL) are examples of commonly used copyleft licenses, but there are several others as well.

Copyleft license terms may be inconsistent with the purchaser’s sales and licensing models, particularly where the purchaser has a “per seat” or “per user” licensing model (or for that matter any licensing model that restricts licensees from making additional copies of the software). Accordingly, such copyleft code, if acquired or licensed in, cannot realistically be distributed by the purchaser or combined in certain ways with the purchaser’s existing products. In situations where copyleft code is located in an acquisition target, purchasers need to develop a plan to have the copyleft code removed, preferably prior to closing the deal.

Pedigree Issues

The second and more subtle OSS issue is related to the “pedigree” of OSS code.³ Even if the OSS code is not governed by copyleft license terms, it may have been developed by a dispersed group of developers who ultimately gave no representations as to ownership and non-infringement of the finished product. The open source “community” development model involves potentially large numbers of developers who are subject to little if any legal oversight and who may be involved in a host of projects, each governed by a different and potentially conflicting license. More importantly, the typical OSS contributor has a day job at a technology company and is likely to have signed a broad employment agreement with his employer vesting ownership of intellectual property (possibly including the contributions made to the OSS project) with the employer. Given what is known about the “community” development process and the lack of warranties on OSS code, the purchaser may have reservations about distributing such code and may want to identify such code during the diligence process for further review.

³ See Terry J. Iardi, *Mass Licensing—Part 2: Open Source Software Licensing*, in Practising Law Institute Patents, Copyrights, Trademarks, and Literary Property Course Handbook Series (PLI Order Number 5939) at 303-4 (June 2005) (“Open source software is risky by nature since it is most often created by a group or confederation of otherwise unrelated developers. There is thus always a risk that the project leader or maintainer does not have all the rights needed to distribute the software under its open source license.”); David Gurwin & David Gould, *Open Source Software: Free But Not Without Cost*, Pittsburgh Technology Council TEQ Magazine (March 2004) (available at <http://news.pgitech.org/teq/teqstory.cfm?id=1148>) (“With those benefits, however, come some significant potential drawbacks [including]... (2) lack of pedigree because the code is developed by many users in many locations over time”).

II. Process Issues with Mergers and Acquisitions Diligence

Locating Open Source Code in Inbound Transactions

Unfortunately, many companies are not aware of the potential issues and risks presented by copyleft licenses. Past experiences and anecdotal evidence gathered from our peers at other technology companies suggest that many small to medium sized companies are just now beginning to institute policies and procedures to address these issues and risks.

Large amounts of OSS are freely and widely available for download in source code form on the Internet and there is nothing inherently different about the way OSS is written. Other than the copyright notices and references to license terms that the original authors may have put in the comments of OSS files (for instance, references to the GPL, MPL or other copyleft licenses), it is very difficult to identify OSS code once it is incorporated into a larger work. Even these copyright and license notices are not a fail-safe way to identify the presence or absence of OSS. The authors of OSS code may fail to include copyright or license notices in their files and subsequent users may strip out copyright or license notices that the original author included.

Automated and Manual Code Review Process

Several companies (including Black Duck⁴ and Palamida⁵) have made significant progress during the last few years creating automated tools that are capable of performing complex searches for OSS code. These companies take advantage of the public availability of OSS code, scouring the Internet and creating a “fingerprint” database of that code. Any code base can then be run through a sophisticated tool that searches for matches with the OSS fingerprint database and identifies areas within the code base that are suspect. These preliminary “hits” can then be reviewed by technical and legal experts to determine whether OSS code has been copied or whether other factors account for the similarities. As these companies fine-tune their technology, companies using these tools will be able to look for fingerprint matches as small as a single line of source code.

Although challenging to orchestrate within the frenetic pace of most software acquisition deals, increasingly purchasers are requiring potential targets to provide (during the diligence process) complete copies of their products in source code form to a designated expert of the purchaser’s choice, for the purpose of reviewing the source code to locate potential OSS and other third party copyright issues. The concept of using third-party experts to review and assess source code has been around for quite some time. Many purchasers routinely conduct a pre-closing review of the target’s source code to ascertain its technical quality and identify potential security vulnerabilities that may lurk within the code. The OSS/copyright review might be viewed as merely an extension (into the legal realm) of these other technical reviews.

The concept of a purchaser (or an agent of the purchaser) conducting a source code level review prior to closing rightly concerns many targets. But with proper contractual

⁴ See <http://www.blackducksoftware.com/>

⁵ See <http://www.palamida.com/>

protections in place, targets can adequately protect their interests and ensure that the purchaser does not gain any access to the code itself, or confidential information relating to the code, prior to closing the deal. Ultimately, these types of source code reviews are beneficial to both parties. Pre-closing assessment of OSS, code quality and security issues can allow the parties to settle a greater number of issues at or before closing. As a result, the parties face less uncertainty about the fate of money tied up in holdbacks or other mechanisms to cover contingencies that may arise after closing.

Experience suggests that using a third-party expert or outside counsel to perform this analysis is advisable. Purchasers and targets can structure the process in such a way that the third party effectively acts as a buffer, conducting the review and passing on a report to the purchaser that includes only the identification of potential OSS or copyright issues, as opposed to any actual code or other confidential information relating to the functionality and operation of the target's products. This type of arrangement is likely to be looked upon more favorably by a target. The purchaser can also structure the process so that the expert's work is protected by the attorney-client privilege, but this would likely preclude sharing a copy of the code review report with the target.

Key Questions in the Acquisition Diligence Process

Aside from these automated OSS fingerprinting technologies, the main tool that purchasers have at their disposal is the diligence process-- in other words, the statements, representations and warranties that a potential target makes regarding the presence or absence of OSS in its products. Some additional information on this process is set out below, including some of the specific questions purchasers can ask of the target.

To make the most of the diligence process, purchasers need to formulate very specific questions and make sure that the contact(s) at a potential target have verified the responses to the questions with their developers (the people who actually wrote the code) rather than relied on generalized statements made by management. It is also important that purchasers begin asking these questions as early as possible, as it often takes time for a target to properly gather the necessary information to respond.

In the end, these questions are intended to highlight a distinction between (1) mere use of software development tools (such as compilers or automated test tools) in binary/object form during the development of a product and (2) incorporation or use of OSS code, modules, utilities or libraries within a product that the potential target ships. The first category (mere internal use of precompiled software development tools) presents very low risks. It is the second category of risks, relating to incorporation or use of OSS in a finished product, that a purchaser would want to investigate fully before proceeding with the proposed acquisition. Listed below are some questions that can be used to help tease out this distinction. In addition, further technical research may be advisable for those not familiar with the process of building or developing software, the distinction between source and binary code, or the definitions of a "library," "module" or "utility."

- ***Question #1:*** *Do any of the potential target's products include code, modules, utilities or libraries that are covered in whole or in part by a copyleft license (i.e.,*

- a license that requires, as a condition of use, modification and/or distribution of such software and/or other software combined and/or distributed with such software be (a) disclosed or distributed in source code form; or (b) licensed for the purpose of making derivative works)?*
- *What specific code, modules, utilities or libraries? (Purchasers should obtain a file by file or module by module listing, including the name of the specific file or module, a brief description of the functionality provided by that code and the name of the applicable OSS license.)*
 - ***Question #2:*** *If yes, does the potential target distribute (i.e., ship) this product or does it merely run internally on the target's servers?*
 - *Since what date has the potential target shipped this product?*
 - *Has the potential target granted rights to (a) disclose or distribute all or part of this product in source code form; or (b) make derivative works of the product?*
 - ***Question #3:*** *Do any of the products of the potential target include code, modules, utilities or libraries that are covered in whole or in part by any other OSS license (i.e., a non-copyleft license)?*
 - *What specific code, modules, utilities or libraries? (Purchasers should obtain a file by file or module by module listing, including the name of the specific file or module, a brief description of the functionality provided by that code and the name of the applicable OSS license.)*
 - ***Question #4:*** *Does the potential target have any policies in place to ensure (1) that OSS is not incorporated into its products without management approval, and (2) that such incorporation is carefully documented? What are these policies?*

Documenting the Results of Due Diligence

Experience suggests that the most useful way to arrange the data that come back from the target is to create a table that lists all of the target's major products or SKU's and then lists potential open source issues. It is important that purchasers record details such as how many lines of code are involved, what specific license governs the code, how it is incorporated or linked, and what functionality that code provides. It is also important to get a sense of whether the OSS code is "core" or "ancillary." If OSS code is found in the core functionality, it may be of great concern, whereas OSS code outside the core may be more easily removed and replaced. As the parties move forward with the potential deal, it will be helpful to have all of this information readily available in chart or "punch list" form so that they can determine appropriate strategies for resolving these issues and check off the issues as they are resolved.

The Impact of the Target's Use of Copyleft Code

Once the purchaser has identified the presence of copyleft code in a target's shipped product (or a product that the purchaser is planning to ship post acquisition), the purchaser must undertake a complex analysis to find out the implications of the inclusion of that copyleft code in the potential target's code. As a general matter, if the target has included copyleft code in one of its products it has likely put itself in the position where it is required to either (1) comply with the terms of the copyleft license at issue or (2) be a copyright infringer for using the code without complying with the copyleft license

conditions. However, there is much debate about what happens when a developer takes copyleft code, combines it with his proprietary code and distributes the resulting larger work, and it is worth elaborating on this point.

When these issues first surfaced and became a topic of discussion among the broader legal community, some commentators suggested that including copyleft code in one's work resulted in the GPL code "tainting" the other code, by the mere act of combining the works and then distributing them. The fact that many people referred to copyleft licenses as "viral" did much to build up the belief that if a code base had ten lines of GPL code in it, it meant the owner could be forced to release the source code and provide intellectual property licenses to the entire work. More recently, a larger group of commentators (including representatives of the Free Software Foundation⁶) responded, suggesting that this proposed outcome could not possibly be the correct result under intellectual property law.⁷ These commentators suggested that the typical copyleft license is merely a conditional license that grants rights to incorporate and redistribute the covered work, but only if the licensee abides by certain conditions (e.g., provide source code and grant intellectual property licenses to the larger work). These commentators suggest that if a developer takes copyleft code, incorporates it into his work and then fails to abide by the conditions, then the developer has merely failed to get a license to the copyleft work and is therefore an infringer. Put another way, these commentators argue that the mere inclusion of some copyleft code does not somehow magically create a contractual covenant on the part of the developer to release source code and provide intellectual property licenses. As an example, these commentators point out that the most prevalent copyleft license, the GPL, is not a contract that creates binding obligations to

⁶ See Bradley Kuhn and Daniel Ravicher, *Detailed Study and Analysis of the GPL and LGPL* (January 2004) (available from the Free Software Foundation)

Section 5 [of the GPL] brings us to perhaps the most fundamental misconception and common confusion about [the] GPL. Because of the prevalence of proprietary software, most users, programmers and lawyers alike tend to be more familiar with EULA's [End User License Agreements]. EULA's are believed by their authors to be contracts, requiring formal agreement between the licensee and the software distributor to be valid. This led to mechanisms like "shrink wrap" and "click wrap" as mechanisms to perform acceptance ceremonies with EULA's.

The GPL does not need contract law to "transfer rights." No rights are transferred between parties. By contrast, the GPL is a permission slip to undertake activities that would otherwise [have] been prohibited by copyright law. As such, it needs no acceptance ceremony; the licensee is not even required to accept the license.

⁷ See Jason B. Wacha, *Taking the Case: Is the GPL Enforceable*, Santa Clara Computer and High Technology Law Journal (January 2005) (discussion of the contract vs. license debate); Phil Albert, *GPL: Viral Infection or Just Your Imagination*, TechNewsWorld Commentary (July 22, 2004) ("Contrary to what some operatives might want us to think, the GPL and other open-source licenses do not have some special magic feature that allows them to infect other software. They have license limitations that apply to derivative works, just like any other copyright license."); Paul Arne, *Open Source Software Licenses: Perspectives of the End User and the Software Developer*, (available at http://www.mmmlaw.com/articles/article_238.pdf) ("The limitations stated in the open source license agreement are enforceable against a user, even when the user doesn't agree to the terms or manifest any intention to be bound, because the rights enforced are copyrights, not contract rights.... Any use of the software beyond the limits or without complying with the conditions set forth in the license exceeds the scope of the licenses granted and is therefore copyright infringement.").

act a certain way (e.g., to provide source code and grant intellectual property licenses) but rather a unilateral grant of permission to use the GPL licensed code provided that you also provide source code and intellectual property licenses to the larger work that you create. There is still some debate around this issue and there are not yet any court decisions dealing with these aspects of the intersection of licensing and copyright law.

In many of the instances where purchasers find copyleft code in the products of potential targets, the target will be unaware of the OSS code and will likely not have made any attempts to comply with the copyleft license conditions. The target accordingly may have been accruing liability as an unlicensed infringer. In other situations, the target may be generally aware of the presence of the copyleft code but not aware of the implications of incorporating that code into its product. Many engineers and managers, and even a handful of lawyers, believe that OSS is synonymous with “public domain” and accordingly believe they can use the OSS code in their products without making any changes to their licensing model and business plan. In these cases the purchaser will want to consider the implications of the target’s unlicensed use of the copyleft code as well as if and how the parties can “remediate” by having the target remove the copyleft code prior to closing.

There may be some instances where the target has knowingly included copyleft code as a separate module or component, in such a manner as to attempt to avoid having to comply with the onerous reciprocal licensing provisions in a copyleft license. In these cases a purchaser needs to determine whether the target’s architectural decisions have in fact successfully avoided the reach of the copyleft license and if not whether the parties can “remediate” the code by having the target redesign or remove the copyleft code prior to closing.

In yet another scenario, the target may have knowingly incorporated copyleft code into its product and then released the entire product under the terms of the copyleft license. These actions may cause the purchaser to reassess the value of the intellectual property in the target asset, since although the target may be able to remediate and remove the original copyleft OSS, it might not be able to “undo” the grant of a broad license to all recipients of the code.

Remediation of OSS Code

If the due diligence inquiry reveals the presence of unwanted copyleft OSS (or other third party copyrighted code), then the purchaser will need to consider whether the code can be extracted and replaced and whether that should be done prior to closing.

There are a number of reasons that remediation is best done prior to closing. First, the target is often motivated during this stage to cure any perceived problems in an effort to bring the deal to a close. Second, the target’s employees are likely best qualified to oversee and perform (at least parts of) the remediation and, depending on the structure of the deal, you may not have access to all of these employees after closing. Third, putting remediation off until post closing may result in a failure to fully appreciate how time-consuming and complex the remediation will be. In the pressure to get the deal done, the

purchaser and the target may not have time to fully appreciate the work required to complete the remediation (including things such as testing). Finally, if the purchaser plans to continue shipping the target's products upon closing, it will face legal liability if it does not either remediate or cure the license noncompliance.

How to Remediate

There is no single way to approach remediation, but there are a number of steps that purchasers should consider. First, they will want to confirm their OSS diligence findings with the target by explaining to the target that they believe certain named files include code governed by certain OSS licenses. If the target does not provide additional information that "explains away" certain OSS issues (for instance a target might show that it has entered into a "proprietary" license with the copyright owner of otherwise OSS code), you will need to consult with the target and explore the feasibility of removing the OSS code. With unexceptional software components, such as compression functionality, the purchaser's engineers may well be able to estimate the amount of resources that would be needed to replace the component, but in many instances, only the target's engineers will have the intricate knowledge of the product needed to make such estimates.

The remediation process should follow industry standard "independent development" guidelines (sometimes referred to as "clean room" development) to ensure that the rewritten functionality does not in fact borrow protectable expression from the OSS files that are being replaced. At a high level, this requirement means that the target engineers who rewrite the functionality should not have had "access" to the OSS source code (by access, we mean that the engineers have not maintained, reviewed or otherwise become familiar with the OSS source code). This ensures that the rewritten functionality is "clean" and not encumbered or restricted by any license terms (such as the OSS license that applied to the original code). Those engineers with "access" to the OSS component or code may "spec out" the functionality, meaning they may prepare a specification that explains to the "clean" engineers what the resulting clean code must do. This specification may be reviewed by counsel or an appropriate expert to ensure that it includes only a description of the functionality and not protectable expression (such as actual source code). When the "clean" developers receive the specification, they should write the replacement code without reference to the original OSS code, and without discussing their work with the specification writers.

Closing Conditions and Verification of Closing Conditions

Just like any other work or activity that the purchaser insists be completed prior to closing, remediation work should be covered by appropriate closing conditions in the relevant acquisition agreement. The parties should consider including within the closing conditions some objective criteria for determining whether the remediation was properly done. These might include both legal criteria (for instance, the purchaser or purchaser's expert re-scanning the code using OSS diligence tools) and technical criteria (successful passage of certain test harnesses to ensure the remediated product works properly).

