From PLI's Course Handbook
Open Source Software: Risks, Benefits & Practical Realities in the Corporate Environment
#5141

10

SEVEN STEPS TO ADDRESSING OPEN SOURCE ISSUES IN SOFTWARE DEVELOPMENT

Stephen A. Mutkoski
Microsoft Corporation

Seven Steps to Addressing Open Source Issues in Software Development

Stephen Mutkoski
Intellectual Property & Licensing Group
Microsoft Corporation

Whether you are in house counsel for a software company or a lawyer representing clients who develop software, it is imperative that you understand the risks that can arise when software developers download and incorporate open source software into company products. This paper is intended to give a broad overview of some of the steps that legal counsel can take to address these risks, including making sure your company has a coherent policy in place to address these risks, that your company employees understand the policy and that you have effective procedures or mechanisms in place to address open source issues as they arise.[1]

1. Create an open source policy for your company

The most important thing you can do is to think through carefully some of the common OSS issues or scenarios that will arise and determine in advance- in a well reasoned fashion- how your company should address or respond to these common issues or scenarios. By way of example, if a developer requests permission to download and incorporate source code governed by the GPL into company products, how would you

[1] This paper assumes general familiarity with open source licensing issues and the common open source licenses such as the GNU General Public License (GPL), Lesser General Public License (LGPL), Mozilla Public License (MPL) and the Berkeley Software Distribution License (BSD).

respond? How would your response differ if the code were governed by the BSD license? What if the request was for permission merely to download and install open source applications for use within the company? It is important to point out early on that there is likely no single "one-size-fits-all" open source policy; each company will likely have somewhat different answers for these common open source questions. Each company's distinct business plan and licensing model (and possibly the company's level of aversion to uncertainty and risks) will dictate the appropriate response for that company to each of these common open source issues.

Why every company needs an open source policy

There are two primary risks relating to open source software that we will focus on here. On the one hand, there are the outbound licensing conditions or requirements that come into play when one incorporates GPL or other similar "Copyleft"[2] code into a company product. By incorporating GPL code into one of its products, your company would likely be required to release source code and grant broad IP licenses to the larger work it creates. The GPL does, after all, condition its license grants on the licensee making the same grants with respect to the larger work it creates. These "reciprocal" grants (and access to the product's source code) could be inconsistent with your company's existing or planned outbound licensing regime.

The second-- less appreciated-- risk relates to concerns about source code "pedigree." Code of unknown or dubious pedigree can transmit with it risks of third party intellectual property claims. Some, perhaps much, open source code is written by a loosely knit group of developers, often with little or no legal supervision to make sure that infringing code or concepts are not being added by the contributors. Many contributors to open source projects have day jobs at companies where they work on related technologies. The employment agreements that are common in the technology industry might well dictate that the "contributions" being made to the open source project are actually owned by the employer. Because intellectual property claims are not dependent on intent or knowledge, your company would be an infringer if unauthorized third party intellectual property were transmitted in the open source code.

Factors to consider when drafting your company policy

The range of responses various companies might have to these risks results from the fact that the intellectual property issues presented can look surprisingly different viewed through the lenses of each individual company's business plan and licensing model. In other words, while one company might care tremendously that incorporation of GPL code into one of its products would require company to make the source code to the entire product available (and grant broad IP licenses to that entire product), another company might be willing to live with these conditions imposed by the GPL. Whether this is the case depends primarily on the extent to which the company derives revenue

[2] "Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it.... [A Copyleft license] gives everyone the rights to use, modify, and redistribute the program's code or any program derived from it but only if the distribution terms are unchanged.... Copyleft is a general concept; there are many ways to fill in the details." See http://www.gnu.org/licenses/licenses.html#WhatIsCopyleft

from licensing its products for a fee to users, because although the GPL does not rule out the collection of a fee in connection with the distribution of a work, it does rule out charging a license fee. Moreover, the broad IP license grants in the GPL (as well as the requirement of providing source code) effectively destroy the ability of a company to prevent others from obtaining, reproducing and redistributing the GPL work to others without paying the licensing fee (in fact the GPL is intended to encourage such redistribution). Even within a company there could be varied approaches depending on the particular circumstances and the particular product. A company might have one set of rules for incorporation of open source code into so-called "core" software assets where intellectual property licenses generate significant revenues but different rules for non-core software assets (where the company might not extract much in the way of licensing revenues).

As a preliminary step to formulating your company's open source software policy, you should first consider your company business model, specifically your outbound licensing plans for software products. You should consider which products the company licenses for a fee to end users, how significantly those license fees contribute to the company's bottom line and whether reducing those licensing revenues (a likely result if the products must be released under GPL terms) will significantly impact the company business plan. If your company generates revenue primarily through consulting or services work or the sale of hardware and your business plan does not include future plans to generate revenues from licensing software, then you might not be averse to including GPL code in company products (and thus broadly licensing your IP on a royalty free basis), because you might not see a reduction in revenues. In fact, there may be situations where giving away software and associated IP (and making it available in source code form) might drive services revenues or sales of collateral company hardware or software products, resulting in an overall increase in revenues. On the other hand, if your company revenues are generated by both services/consulting work and software licensing fees, you will need to consider the potential implications on licensing revenues of particular products, particularly those "core" products that account for high percentages of software licensing revenues. If your company revenues are generated primarily by software license fees, you will likely want to be more conservative, carefully scrutinizing each proposed instance of open source use to make sure that the significant revenues from licensing fees are not jeopardized.

Some specific scenarios to consider

In drafting your company's open source policy, you will want to consider a range of potential scenarios that developers may bring to you. Obviously one of the most important scenarios for you to consider is the one in which your developers ask to include open source code in a company product. But you will also want to consider whether or when (a) company code should be released under an open source license, (b) employees should be allowed to participate in external "community" projects and (c) your company will set up external community projects for its products, including taking back in community contributions.

Depending on your company business and licensing models, you might ultimately make the decision to allow developers to include open source code into a particular company product. But given the fact specific nature of each inquiry, the complexity of open source licenses and the unsettled questions concerning what product architectures might require compliance with certain open source license provisions, you will likely want to have a policy that requires approval of each specific instance where open source code is incorporated in a company product. For example, depending on the open source license it might make a difference if the employee (a) cuts and pastes the code into company product (b) statically links to the code or (c) dynamically links to the code. And, your developers might interpret an open source license quite differently from how you would as legal counsel. Overall then, it might make sense to have a policy that requires approval prior to any incorporation of open source code into a company product, with each specific request being reviewed by an appropriate legal or business decision maker.

The person or persons responsible for reviewing requests for permission to incorporate open source code should be guided by a much more detailed policy document. This document should reflect the company's own business plan and should point out acceptable and unacceptable requests. For instance, one company might decide that its reliance on end user licensing of its products cuts against using any "Copyleft" code, but that with appropriate review it can get comfortable with "pedigree" risks from non-copyleft code. Another company, while generally restricting incorporation of GPL licensed code might allow incorporation of code licensed under the Common Public License (CPL) or Mozilla Public License (MPL) if certain product architecture requirements are met (i.e., files are segregated). In the end, this detailed policy document should reflect what you think each open source license means (when it comes into play, what it requires you to do, etc.), what products your company needs to continue licensing for a fee as well as how you feel about potential pedigree issues with open source code.

In addition to addressing open source code "incorporation" as discussed above, you might want to include in your open source policy your company policy for internal use of open source products. There are a number of useful development tools released under open source licenses that your developers may want or need to use in developing your products. The vast majority of these tools are used only in the development process and they in no way are incorporated into the shipping product.[3] Your policy can either explain that tools may be used without limitation provided they are not in whole or in part incorporated into a company product, or set out an approved list of tools that you have confirmed do not cause code to be incorporated into the product under development.

---

[3] A tool called Bison is one such exception. See http://www.gnu.org/licenses/gpl-faq.html ("Some programs copy parts of themselves into the output for technical reasons--for example, Bison copies a standard parser program into its output file. In such cases, the copied text in the output is covered by the same license that covers it in the source code. Meanwhile, the part of the output which is derived from the program's input inherits the copyright status of the input. As it happens, Bison can also be used to develop non-free programs. This is because we decided to explicitly permit the use of the Bison standard parser program in Bison output files without restriction. We made the decision because there were other tools comparable to Bison which already permitted use for non-free programs.")

You will also want to make sure that your policy addresses whether and when employees may release company code under an open source license. Many developers are actively involved in open source projects outside of work and may not understand the implications of releasing company code under an open source license. If the code includes valuable company intellectual property, the company may lose a competitive advantage in the marketplace, since release under most open source licenses would result in the grant of broad licenses to this intellectual property. Again, the specific instances where a company might want to prohibit or allow the release of certain of its code under an open source license will vary from company to company, but each instance should be reviewed by an appropriate legal or business decision maker (namely one who has a solid understanding of the company intellectual property policy and its licensing plans).

Whether you know it or not, your developers are likely involved in various "community" projects and activities. Community is a broad term used to refer to newsgroups, Sourceforge-like sites and other areas where developers "gather" and collectively work on building code or solving more general problems. You should consider whether these activities might expose your company to risks. For instance, if you have a product that you license to end users, you probably want to restrict how your developers who work on that product get involved in community projects that relate to the same or similar functionality. If you impose no such restrictions, your developers might wind up contributing valuable trade secrets, copyrights or patents to the community project. Although you may want your employees to be involved in certain aspects of the community, you likely do not want them to contribute all of the know how and other intellectual property that makes your product superior to the similar open source product. In addition, your developers might bring something back from the community and incorporate it into one of your products and, as discussed above, not realize the implications of doing this. Depending on your specific situation, you will likely want your open source policy to explain the possible risks of participating in community projects, whether or not on company time, and at a minimum require approval for involvement in projects involving functionality that is similar to what the employee develops at your company.

A final area that you should consider addressing in your open source policy concerns when or whether your company will set up or "sponsor" community projects and what your company will do with the fruits of such a project. While a community project might seem like an obvious choice for one or more of your company products (to harness free developer resources), there are a number of risks and administrative burdens you should consider before you undertake such a project. For instance, just as there are potential pedigree concerns with code your developers download from the internet, there are such concerns with code added in a community process. You would want to get some sense of who your contributors are, confirm they have intellectual property rights in the contribution sufficient to allow you to incorporate it into your company product (the Free Software Foundation in fact does just this) and possibly have them assign their rights in such contributions to ease future administrative burdens.

2. Education, education and more education

Once you have formulated your company's open source policy, it is essential not just that you convey that policy to your employees, but that you give them a basic understanding of the issues or concerns that open source software can raise. You should consider how you can tailor this training to specific groups within the company. For instance, developers, architects, managers, executives and legal personnel all have different technical and legal backgrounds and would benefit from a well-tailored training. Legal personnel might need a training that explains concepts such as dynamic and static linking, or even some more basics of software development such as the distinction between source and object code.

You should also consider how you can orient the tone of the training in such a way as to encourage a partnership between you and the developers, who might be inclined to resist what they see as legal meddling into the product development process. Explaining the possible conflict that developers might create with the company's licensing plan by incorporating open source code into company products is one important means of relaying to your developers the need to be on the look out for open source issues.

Finally, open source training is one that is usually best done in person and with a format and time allotment that allows for real time Q&A. The Q&A helps insure that your audience gets the message you want to convey, but it can also help you adjust your approach in future training sessions. You will likely find that your materials need to be revised or expanded to cover certain frequently asked questions or hard to understand concepts.

3. Create a mechanism for answering developer requests

As discussed above, it is quite likely that your open source policy will include at least a few areas where developers are required to seek approval, for instance to incorporate open source code into a company product, to contribute to a community project, to release company code under an open source license or to set up a community project. You should put a mechanism in place to make sure that these requests are taken care of early in the development process to avoid last minute delays. If open source code is introduced into your code base without approval and not located until just prior release, it could hold up shipment of the product. It is normally easier to review and resolve these issues early in the product development cycle before ship deadlines are rapidly approaching.

You should also consider creating a request mechanism that helps route requests to the appropriate decision maker and at least for some initial period consider centralizing the task of reviewing and responding to such requests. Many companies find that during the initial period after implementing and open source policy, it often makes sense to consolidate authority for reviewing open source related requests into a centralized group

or committee, since a committee can draw upon each successive determination and insure that the requests are resolved in a manner consistent with the company's long term goals.

4. Carefully document open source use

To the extent you approve certain requests to incorporate open source code into company products, you should make sure to carefully document that use. At a minimum you should make sure your developers preserve any copyright notices in the code and archive an electronic copy of the open source license attached to the code. It is also helpful to have a system that connects all of this information together, for instance accurately tracking precisely which code is subject to which open source license. Such a system makes it easier to check that you are complying with open source license terms and it makes you more agile if your product is part of an M&A transaction and you need to quickly get a potential licensee or buyer information about the product's open source dependencies. You should also consider how you will flag and comply with any obligations imposed by the open source license, for instance the obligation to carry forward attribution or to provide source code.

5. Find your experts

Although the majority of open source requests will be ones that you and your extended team can address, there will occasionally be questions that require outside legal and technical expertise. Frequently these complex questions will be related to how alternative product architectures might or might not bring certain license provisions into play. Given the high stakes of these decisions, it makes sense to involve technical and legal experts who routinely analyze these problems and can draw upon a wealth past experiences. You will likely not realize that you need such outside expertise until it is almost too late and you have product build deadlines or ship dates looming on the horizon.

6. Tune up your M&A diligence practices

If you are concerned about what open source code your own developers might put into one of your company products, you should be equally concerned about open source code that might be present in code that your company acquires or licenses in. But, how can you spot open source code in a potential acquisition target? Unfortunately at the present there is no magic tool that you or the potential target can use to find open source code in their products. Open source code is freely and widely available for download and there is nothing inherently distinct or different about the way open source code is written. Other than the copyright notices and references to license terms that the original authors may have put in the comments of these files (for instance references to the GPL, MPL or other Copyleft licenses), there is practically no way to identify open source code once it is incorporated into a larger work. Even these notices are not a guaranteed way to identify the presence or absence of open source code. The authors of code may fail to

include copyright or license notices in their source code files (this after all is a convention and not a requirement of copyright law) and subsequent users of these files may strip out any copyright or license notices that the original author included.

Accordingly, the main (and sometimes only) tool that you will have at your disposal is the diligence process-- in other words the statements, representations and warranties that a potential target will make to you about the presence or absence of open source code in their products. To make the most of this process, you need to be very specific in your questions and you need to make sure that your contact(s) at the potential target have verified the responses to your questions with their code writing developers, rather than relied on generalized statements made by managers. The following questions provide a good starting point:

> 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; (b) licensed for the purpose of making derivative works; or (c) redistributable at no charge)?
> What specific code, modules, utilities or libraries?
>
> Question #2: If yes, does the potential target distribute (i.e., ship) this product or does it merely run internally on target's servers)?
> Since what date has the target shipped this product?
> Has the potential target granted rights to (a) disclose or distribute this product in source code form; (b) make derivative works of the product; or (c) redistribute the product at no charge?
>
> Question #3: Do any the products of the potential target include code, modules, utilities or libraries that are covered in whole or in part by any other open source license (i.e., a non-copyleft license)?
>
> Question #4: Does the potential target have any policies in place to insure (1) that open source code is not incorporated into its products without management approval and (2) that such incorporation is carefully documented? What are these policies? How does the target track open source license obligations and insure that the target complies with these obligations?

7. Audit your products before you release them

As you near completion and release of your products, you should consider performing an open source audit, to confirm the documented instances of open source use and gain some assurances that developers did not incorporate additional open source code without obtaining an approval. If you have kept good records of the instances where open source incorporation was approved then the first part of your audit will be easy and

straightforward. You will also want to reconfirm that you have a plan in place to comply with an open source license obligations, such as attribution or source code distribution requirements.

You might want to take additional steps to insure that your records reflect all instances of open source incorporation into the product. This typically involves a meeting with the development team to confirm that each member of the team followed the company open source policy.

Finally, you might want to consider using some type of automated tool to scan your source and/or object code prior to release. There are currently two different types of tools. The more basic tools merely perform basic text scans on file headers and comments, searching for copyright strings or license names that could indicate the possible presence of OSS code. It might surprise you to that many companies with well thought out corporate OSS policies none-the-less locate instances of OSS in their products by performing such rudimentary searches. Successfully locating OSS code with these basic string search tools is however dependent on employees retaining (or copying in) copyright notices or license terms. To the extent employees have removed copyright notices (or to the extent there were no notices in code in the first place), this type of code scanning tool will not locate instances of OSS code.

More sophisticated tools are now in the works that offer the promise to find substantially more OSS code- including code that has no copyright notices or license terms attached- by "fingerprinting" known OSS code bases and making intricate comparisons to an internally developed code base.

SAMPLE DILIGENCE QUESTIONNAIRE

Overview:
We would like to ask a few questions about incorporation of open source software (OSS) into your product and/or code base. The questions are intended to differentiate between two general categories of OSS (1) so-called "Copyleft" code and (2) non-Copyleft code.

The first category of code requires as a condition of use and distribution of the code that you disclose (all or a part of) the work containing the licensed software in source code form and provide broad IP licenses to (all or part of) the work. Such licenses are often described as being "viral" or "reciprocal" or "free." The conditions imposed in these licenses conflict with the traditional proprietary software business model that is based on binary only code distribution and granting of limited IP rights pursuant to an End User License Agreement. Examples of Copyleft licenses include the General Public License (GPL), the Lesser General Public License (LGPL) and the Mozilla Public License (MPL).

The second category of code does not have the same source code and IP license conditions, and instead typically conditions use and distribution of the code on providing attribution of the copyright holder. While these licenses do not conflict with the traditional proprietary software business model (as Copyleft licenses do), they typically disclaim all warranties as to ownership and non-infringement. Examples of non-Copyleft licenses include the Berkeley Software Distribution (BSD) or the Apache licenses.

Questions:
1. Copyleft Code: Does your code base 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; (b) licensed for the purpose of making derivative works; or (c) redistributable at no charge)?

If you answered yes:

- Please identify what specific code, modules, utilities or libraries, what functionality they provide and what particular license covers that code.
- Have you previously distributed (i.e., shipped) this product or does it merely run internally on your servers?
- Since when have you distributed this product?

2. Non-Copyleft Code: Does your code base 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)?

If you answered yes:

- Please identify what specific code, modules, utilities or libraries, what functionality they provide and what particular license covers that code.
- Have you previously distributed (i.e., shipped) this product or does it merely run internally on your servers?
- Since when have you distributed this product?

3. Company Policies: What policies do you have in place to insure that OSS is not incorporated into your code base by your developers? Is there a process that employees must undertake to obtain management approval prior to incorporating OSS into your code base? What procedures do you have to insure that such incorporation is carefully documented? What training do you have to insure that developers are aware of these policies? When were these policies instituted?

Open Source Software: An In House Perspective

Stephen Mutkoski
Microsoft Corporation
Intellectual Property & Licensing Group

Microsoft

Overview

- Case Studies
  - Internal development issues
  - Releasing company code under an OSS license
  - Community Issues
  - Outsourcing and M&A Issues
- Appendix: Business Model Observations
  - Subscription Model: Red Hat & Sun
  - Dual Licensing: MySQL

Microsoft

Internal Development Issues

- Scenario #1 "Code Incorporation"
  - Developer requests permission to cut and paste 5,000 lines of BSD licensed code and link to several functions in a GPL library
  - Issues: BSD code "pedigree"; GPL linking and possible license requirements
- Scenario #2 "Internal Tools Use"
  - Developer requests permission to use Bison, a parser generator, and NUnit a testing tool in developing a product that company plans to distribute
  - Issues: Bison includes some of itself in output; NUnit does not
- Lessons
  - Effort required to distinguish between two scenarios
  - Code incorporation is likely time consuming one-off inquiry
  - Code pedigree risk is often overlooked

Releasing Code under an OSS License

- Scenario: Developer requests permission to release Company code under an OSS license, suggesting the GPL as the license
- Issues
  - Loss of trade secrets?
  - Scope of IP rights granted by GPL?
  - Appropriate license for licensees?
- Lessons
  - Source licensing is not one size fits all proposition

Microsoft

Community Issues

- Scenario #1 External Community Participation
  - Developer requests permission to participate (i.e., contribute to) an existing external community project
  - Issues: “Leakage” of company IP
- Scenario #2 Establishing Company-Sponsored Community
  - Developer requests permission to set up a company-sponsored community site in connection with the release of company source code, to allow others to contribute to, improve and modify the company code; developer wants to take back in contributions and roll them into company release of product
  - Issues: “Pedigree” of contributions
- Lessons
  - Very difficult to limit how developer contributes in Scenario #1; Oversight is difficult; Possible that developer could contribute core company IP if proper screening/walling off is not done
  - Administrative and screening process for Scenario #2 is time consuming; impossible to completely eliminate pedigree risks

Microsoft

Outsourcing and M&A Issues

- Scenario #1 Work for Hire
  - Company hires third party to write software component for incorporation into company product; Third party includes BSD and GPL code
  - Issues: Risk that you will be forced to chose between pulling the OSS code and complying with the OSS license; potentially costly rewriting process
- Scenario #2 Company Acquisition
  - Company considers acquiring software company, including code assets and engineers; Target has used GPL and BSD code in developing its products
  - Issues: Risk that you will be forced to chose between pulling the OSS code and complying with the OSS license; potentially costly rewriting process
- Lessons:
  - Agreements with third parties should include OSS warranties
  - Level of sophistication varies across industry
  - Use diligence process to tease out OSS use in specific products and company policies on OSS use
  - Education of "new" employees from target

Microsoft

Valuation Implications

- "When IBM acquired Think Dynamics, a painstaking manual examination of its code revealed 80 to 100 examples of open source code that Think Dynamics programmers had passed off as their own. As a result, the price of that company went down from 67 million dollars to 46 million--not a happy moment for its owners and shareholders, I'm sure."
  http://www.oreillynet.com/pub/wlg/4291#infringers

Microsoft

Buyer's Remorse

- "The Linux software in the router is distributed under the GNU General Public License (GPL), which the Free Software Foundation created in 1991. Under the license, if you distribute GPL software in a product, you must also distribute the software's source code. And not just the GPL code, but also the code for any "derivative works" you've created--even if publishing that code means anyone can now make a knockoff of your product. Not great news if you're Cisco, which paid $500 million for Linksys. In Cisco's case, it's even trickier, because the disputed code resides on chips that Linksys buys from Broadcom. So now Cisco is caught between the Free Software Foundation and one of its big suppliers."
  http://www.forbes.com/home/2003/10/14/cz_dl_1014linksys.html

Microsoft

Appendix:
Business Model Observations

Microsoft

Problem: Generating a per copy fee with GPL Software

- The GPL precludes traditional end user licensing model
  - GPL effectively only allows for charging a fee for the “first” copy
  - You cannot restrict (1) end users from making and installing additional copies or (2) distributors from selling copies of your product
  - No “piracy” mechanism permitted under the GPL

Red Hat's desktop efforts

- Who will buy RH 9.0 for $80 when you can download for free or even buy complete boxed set of CDs for $10?
- RH attempted to stop bulk copiers/distributors with trademark law; Claimed distributors had to remove all RH logos, icons and other marks, and that software could not be marketed as Red Hat
- Lessons
  - Fedora.org now develops/maintains RH's consumer version of Linux with support from RH
  - Core RH focus is enterprise server market, not desktop

Microsoft

The Red Hat Subscription Model

- RH Server code is distributed under the GPL
- Subscription Agreement that entitles licensee to ongoing stream of security and performance enhancements
  - Requires payment of per server fee
  - Includes audit clause typical of traditional EULA
  - Terminated if GPL code is modified
- Enables distributor to receive “per copy” fee that looks a lot like a software license fee

Microsoft

Sun's Java Desktop System

- JDS distributed either in "static" form for one-time fee or "dynamically" for annual per user fee
- Sun employees recently publish blog about the economics of subscription model

Microsoft

Dual Licensing

- What is it?
  - Releasing a same or similar code base under two different licenses
  - Usually one license is open source and the other proprietary
- Who can dual license code?
  - Usually only the copyright holder
  - The copyright holder can release its code under numerous non-exclusive licenses
  - The recipient of GPL code cannot, of course, release that code under any license other than the GPL

MySQL's Dual Licensing Scheme

- Why it works?
  - It's really a great marketing tool, an opportunity to generate leads for the "full" version
- What's left to sell, who would buy it?
  - MySQL isn't a product that installs out of the box, significant services and support
  - MySQL hires the good MySQL developers
- Don't try this at home?
  - Complexity of product and nature of competition in product space limit utility of dual licensing
  - Recent comments by MySQL about when commercial license is required (see http://www.ofb.biz/modules.php?name=News&file=article&sid=325)

Microsoft