

ASSOCIATION OF CORPORATE COUNSEL

TITLE: Open Source in the Real World: Beyond the Rhetoric

DATE: January 15th, 2008

PRESENTED BY: ACC Information Technology Law & eCommerce Committee

SPONSORED BY: DLA Piper

FACULTY: Kat McCabe, VP & GC of Black Duck Software, Inc.
Maureen Dorney, DLA Piper
Gemma Dreher, BAE Systems

MODERATOR: Maureen Dorney, DLA Piper

Operator: Just a reminder. Today's conference is being recorded.

Female: Welcome to this ACC Webcast.

Maureen Dorney: Welcome, everybody. My name is Maureen Dorney. I'm a partner at DLA Piper in Silicon Valley and we're very pleased to be able to speak to you in this Webcast.

With me I have Kat McCabe and Gemma Dreher. I'll read bios in a minute.

A couple of housekeeping before we begin. We will be speaking for approximately 50 minutes and then taking about 10 minutes of questions at the end of the hour. To submit a question, what you do is you go to the questions box in the lower left-hand corner of your screen, type in a question, and click "Send."

I've also been asked to remind you to fill out the Webcast evaluation, which is in the links box, and also that a copy of our – sorry – PDF copy of our slides is available as item four in the links section of the Webcast.

I'd like to first introduce my fellow panelists.

Gemma Dreher is senior legal counsel for the Electronics & Integrated Solutions operating group of BAE Systems. She provides advice and services concerning the legal matters of the operating group and represents – excuse me – the operating group in advisory proceedings. She serves as a consultant to management and external spokesman for the organization on major matters pertaining to its policies and objectives.

Gemma has developed a litigation risk assessment plan for the legal department, and has also developed an outside counsel protocol and internal file management system for the legal department. She speaks frequently on the subjects of intellectual property management, licensing technology to the federal government, and open source software licensing. Gemma joined BAE Systems in 2002. Prior to that, she was vice-president and general counsel for Pragmatech Software. She's been practicing law for 17 years.

Also with me is Kat McCabe. Most recently, Kat was vice-president and general counsel of Black Duck Software based in Waltham, Massachusetts. In her role at Black Duck, Kat was a frequent public speaker on open source licensing and legal issues.

Prior to Black Duck, Kat spent five years as general counsel of Kodak Imaging Network, formerly Ofoto, based in Emeryville, California. And prior to Ofoto, Kat was in the licensing group of Fenwick & West. Kat received her BA from Brown University and a JD from Harvard Law School.

I have been practicing here in Silicon Valley for the last 20 years, last 13 years at DLA Piper and as predecessor from Gray Cary, specializing and representing public and private technology companies in all management intellectual property technology licensing, open source privacy and digital media, and any commerce issues.

So with that introduction, I'm going to move on, talk a little bit about what we're talking about today. This is a – this conference is a reprise of a presentation we did at the annual meeting in Chicago last fall. It was pretty favorably received, and we were asked to redo it.

And what we tried – what we've tried to do in this set of slides and materials and the comments we're going to make is really focus on the fact that there is now widespread availability in use of open source software. It's here to stay in various forms. And it makes it pretty critical for corporate counsel to understand issues and best practices, and to really develop practical solutions in how they're going to handle open source in the context of the specific facts of their company.

So we're trying to get practical today, not theoretical. And what we're going to do is focus today on management of open source in the context of development of software in the procurement chain and in due diligence and M&A.

Gemma will first – speak first on the development, and I will cover procurement, and Kat will cover M&A from the buyer perspective. And Kat will also talk a little bit about the GNU GPL free license. And after that, then we will, for the time remaining, review your questions.

So I am going to transfer the presentation over to Gemma and let Gemma take over.

Gemma Dreher: Hello everyone. I'm going to speak about open source and the development process, and the first thing I want to mention that I think is very important when considering how to handle open source software is what the company's open source strategy is.

In my involvement in this area, a lot of the way you answer these questions depends upon whether or not your company wants to be a key player in the area of open source in the open source marketplace or is using it as a competitive advantage or is simply grudgingly acknowledging that open source is now a development model that needs to be contended with. So, for each of these areas, the approach that you take is highly dependent upon the strategy that your company has.

Relative to internal policies, there are several things that you must consider when developing policies or procedures. You need to look at what your use will be for internal purposes, what it will be for external purposes, and then what you will require as far as your employees, your software developers, as far as contributing back to the open source community.

I have seen many policies and procedures where each use of open source software needs to go through an approval process. So, it is rare in many instances that something will be approved for any conceived use. If there is a specific use that's approved, and then the open source wants to be – there's a desire to use the open source again, another approval is required.

And this will depend, too, upon what the usage is. If it's just for internal use, you may see a more liberal approval process as opposed to external use, and it also, again, will go back to the strategy for the company.

When looking at developing policies and procedures, you – there are various approaches that have been taken or can be taken in this area. There are companies that do this in a centralized fashion by committee so that all requests for open source usage are channeled through either an IP licensing department or some centralized committee that reviews each source – each request for use of open source.

There is also the possibility of some pre-approval or disapproval of certain licenses. Of late, I've talked with colleagues in the industry about blanket approvals for certain types of licenses depending upon the usage. You know, if it's internal use, typically you'll have broader approval. If it's external use that might be reviewed up through a committee with less liberal or more conservative approaches taken.

And then the third option for managing is just having review on an individual case-by-case basis. With each of these various approaches, the most important thing from my experience, and again talking with colleagues, is to educate developers and others in this area around procedures and risks.

As I speak on this subject to both the legal community and the engineering community, it becomes apparent that many folks still don't appreciate the fact that open source software is not software that's in the public domain. It is oftentimes difficult to find licenses associated with a particular piece of code, and therefore, particularly the engineering community that I've worked with, assumes that this is either public domain software or that it comes without any restrictions or any associated risk.

And so, education is an extremely important component of managing the risks of open source and open source – the usage of open source in the development process.

So, essentially, as far as best process is concerned, when considering how to mitigate risks in this area, requiring a review of approval before – prior to any use or integration of open source software into the code base is critical. It's important to look at the applicable license and understand what the source of the code that is being used is.

And there are methods for determining the maturity of the code, the viability of the code. I have found, in my work in this area, that it's extremely important for technical and legal personnel to have strong communications and for – and I – in my role for my company, I defer and rely upon my technical experts to tell me some of the issues that might exist with a particular piece of code as we contemplate integrating it into our code base, either for internal or external use.

So it's very important for that synergy and that communication to exist between technical and legal personnel. And again, not being a technical expert myself, I rely on my engineering community to review the code and look at developer comments. And then outside resources are also available for auditing and identifying open source that may have already crept into the code base.

So the last, and again, as important piece of this as education, is documentation of the use of source code. I've talked with a number of different companies, and we've employed some of these techniques internally at my company is to understand where open source is in the code base, where it has already been used in the development process, what the location of that code is, the name of the individual that brought it in to the company, the intended use of it, of what version has been brought in, what has happened to that, what is – how have we controlled the usage of that open source code once it has been integrated into the code base or has been used for internal purposes.

And also what is the applicable license. It's very important to understand what the license is. I mentioned previously the fact that oftentimes, it's difficult to find a license when a piece of open source code is being brought in. In my company, we place that obligation on the requester to be sure that they have submitted the applicable license with their request of approval, and that as the requester, they also understand the obligations that they are submitting to in their desire to use the open source software.

And then, an equally important piece of this is to ensure that we are complying with all the obligations of the usage of the open source. So, have we ensured that we don't run into any source of licensing conflict, have we looked to ensure that the particular license that we intend to use complies with all of our other policies and procedures that are imposed upon the development of software. And, most importantly, have we captured all of the usage and mitigated or cast any risk associated with integrating it into a deliverable, whether that's an open source deliverable or some proprietary code. So, those steps are also extremely important.

So, in summary, I think that it's important to understand that and make the using community understand that there are risks associated with open source software. And that can be done most effectively by education.

I've had instances where I've spent a good deal of time talking about the risks associated with pulling something off the Internet without understanding what the obligations of using it are. And even after spending an hour more of that talking to very bright engineers, I'll have someone ask me, "Well, you know, just even though you say that, because I can get it for free, doesn't that mean that I'm free of any associated risks?" So, that's something that really needs to be drummed into the using population's head, if you will – their collective head.

And then, as I said – in closing, ensuring that you're complying with the license is looking at any conflicts of licensing and understanding what your strategy is and how you can comply or fulfill that strategy through the approval of various pieces of open source code.

So I'm going to pass it back to Maureen at this point.

Maureen Dorney: Thanks, Gemma. So, as we move on to the next topic, procurement, I know we're getting a few questions, and we will try to answer as many of them as we can at the end.

Procurement is a natural extension of the development process, but what it requires, what it gets us into, is an additional level of complexity. Because when you look at the potential, both upstream and downstream sources in the open source procurement ecosystem, you can have it not only from your internal development, but you can have it from third-party developers, including offshoring vendors and offshoring companies. You can get it from enterprise software vendors.

This can be a model both upstream and downstream. You know, you can have ASP or software service providers who have to figure out their open source strategy and whether it's going to be different. It's really because there is a – open source – software service provider (with) their open source strategy changes. They're also going to have an enterprise component.

And probably most – the most complex is the OEM relationships, when you have a software component that it's a critical OEM – your model is OEM distribution and your product becomes or it incorporates critical components from other people. And then both – also downstream is a critical component of other people's software. You really get into the issue of inconsistent policies. I think the VAR and the ISV is really an extension of the OEM relationship.

And I have even found, in my negotiations on behalf of clients, that one of the additional complexities is you often can get different divisions of the same technology company employing –

deploying conflicting policies. You can have one division that has a very (extensive) – really been designed as a large OEM procurer that's been designed to take advantage of open source software and build support with the open source community. And they have one policy of how they handle this.

And then they – you have another that is pure proprietary. And they have – they have a different policy which can be complicated, let's say office strategy. Also, the other thing that needs to get woven in when you're setting out a new strategy and best practices, if you're procurement or your downstream channel strategy is whether or not there's a dual-source model. If there's – if there's potentially a proprietary license available for extra charge, would that be a better route, given what your downstream suppliers are asking for?

So I'm going to flip to the next slide. And that's really the ecosystem.

The issues when you're formulating an open source procurement strategy is it's got to be parallel with and compatible with most of your internal development and downstream licensing strategies. So, you have to look at you channel and what they are going to require. You have to (simulate) through the process that Gemma talked about for internal development.

You have to understand your software architecture. If you do not understand your software architecture and then how each piece of potential open source code is going to be added on to your architecture, you're not going to understand the implications that you need to achieve. And you're not doing licensing basics 101.

But as everyone know, open source licenses go from very simple notification requirements from a BSD license to – all the way to the other end to the GPL, LGPL, GPLv2 and GPLv3 licenses, which have much more requirements passed on downstream for that component of the code, and also what the consequences are to the rest of the code base if you architect in that GPL – more demanding GPL license, if you have certain types of connections.

So you not only have to understand the license and understand the functionality of the software. You really have to understand the architecture and have a strategy for how you're going to handle that. You have to make sure that your internal risk management team has come to a position on the incorporation of open source code as part of the procurement strategy and in terms of the impact on warranties and indemnities.

And that becomes very important in downstream licensing. I can't tell you the number of times I've seen companies take on open source code because it provides a critical or important component to them.

But of course, those come without generally warranties and indemnities. You can get some limited, at least servicing – service coverage under some of the open source licenses, but many of them come with no warranties and indemnities.

But if that becomes a component of your product, it is very difficult these days, in my experience, and should (take a full) commercial context to get to the next stage with your important OEM customer or enterprise customer. If you try to say in your agreement, "I'm carving out these aspects of my product because they are subject to open source licenses," (don't wonder they're not) giving you any warranties or indemnities. So, you have to really think about, in your negotiation strategy and your contract strategy if you're including open source software, how you're going to reflect that.

And then one of the devil-in-the-detail issues is if you've incorporated open source software into your – into your code, make sure you're passing on the licenses and the proprietary rights notice in a way that it is conforming. And that is – another big place that companies get into trouble is

they have what it – they often – well, I've seen demand letters coming from open (status) license source or Free Software Foundation or someone in the open source movement.

Very often, the first thing they're picking up on is that the – you're not clearly disclosing in your software licenses your read me files. However, your license terms are getting passed on downstream that certain software is not subject to the proprietary license, but instead it's subject to this other license. And getting that paperwork and those details right is work the engineers building the read me files often have to participate, and that they don't like to.

But one of the things you should do when you're building in open source software either in you development or you procurement is think about that next stage and make sure you're passing down through the channel all the notices and the licenses in a way that clearly label and disclose the software.

And then, again, you can always consider using dual source options where appropriate, where maybe you can get out of some of the more onerous open source obligations to the extent you think you have to tightly architect that software in by seeing if there is a higher cost purchase on a dual source model.

I think, generally, what Gemma covered for open source policies and approval structure for internal development should extend to procurement. They should be integrated and, you know, note as I go on to this next slide, procurement partners can have radically different policies.

What I am seeing right now, in my experience, is that – is that the procurement partners have very, very different strategies, and it causes all kinds of consternation when you actually get down to doing the business deal. And I'm going to give you three example procurement clauses.

The first one was actually in the context of a spinout. And in this context, the company that was spinning out this code to a new company was very, very nervous. And the license – they had a clause in their license agreement that you'll see on your screen that was just un-administrable in its breadth. And I'll just – I'll just try to highlight.

They said the "Company shall not combine or distribute the Source Code with any Publicly Available Software." So, that restriction was absolute prohibition of using any open source or free software. And it didn't really matter whether or not it could be combined and distributed in a way that wouldn't cause any negative consequences for the proprietary software that was being licensed here.

Then it went on to define publicly available software as any – each of any – and I won't read the whole thing, but I'll just underline – include software "that contains, or is derived in any manner (in whole or in part) from, any software that's distributed as free software ..."

Again, from a legal context, "contains or is derived in any manner." What does it mean to be derived in any manner? That was not clear. And for this to be workable for the spinout, this language had to be substantially revised before we could go on and build a company.

The next example, the second example of the three examples, was something that I thought was actually really quite reasonable. And this was something that – this was a very large hardware and software company, and it really focused – the clause really focuses on what is reasonable that you want to be thinking about if you are OEM-ing in a PC software that is going to have downstream consequences.

What they asked the licensee to do is first give them a clear list of all the open source technology. You know, that's A, and includes GPL.

Then B, which I think is great, probably better than my ability to draw these lines, is a description of how the open source technology is incorporated into, because you need to know what it is. And then you need to know how it interacts so you can understand the consequences.

And then the third thing they wanted is a copy of the license governing the use and distribution of the open source technology.

And then they went down below to say that the licensor agrees to fully cooperate with the licensee to ensure compliance.

So what they said – what they really did was say, “Hey, going into this contract, I want full disclosure. I wanted – of the things that – all three things that matter, from you knowing how to – how I will be impacted by your use of open source software. And I want you to work with me.” And I thought that was much more reasonable than these broad prohibitions.

The third example that I’m going to show you actually is the polar opposite of the first one. In this example, the OEM, in exchange for getting a license to the software code, was requiring the licensee to give the OEM control over whether or not the – what was proprietary software could be released under any fashion the licensee may choose to, including but limited to open source or community licensing except BSD, to reproduce, et cetera, et cetera, the software in source code and object code form. So this was giving the OEM, who had a robust open source policy, the right to choose, at some point in the future, to make the software that was being OEM-ed open source. So, that was – that was somewhat of a surprise and the complete opposite situation.

So I think, with that, I’ve come to the end of the procurement slide. And I’m going to pass it back to Kat to talk a little bit about the due diligence M&A process.

Kat McCabe: Great. Thank you, Maureen, and good afternoon everyone.

In the M&A context, the advent of open source has changed how buyers think about their own due diligence and what code they bring in-house through acquisitions. Traditionally, buyers learn about a target’s code base by reviewing contracts and interviewing the engineering management team. And while that’s certainly still useful, it doesn’t produce the whole picture anymore.

A code, of course – open source code can be downloaded off the Web without any contractual record. So, contract review won’t happen in that case. And because of the easy access to code online, the target’s management team may not be aware of code that their employees have brought in. So, in that case, of course, interviewing the management team won’t help.

So, buyers know that’s it easy for the target to lose control over what code is coming in-house. And they also know that the traditional due diligence won’t give them all of the information they need. So in response, buyers that acquire technology companies now routinely analyze the actual contents of the target’s code base.

So in a few minutes, I’ll talk a bit about how that works as a practical matter. But before I do, let me talk more specifically about the buyer’s concerns.

First, the buyers concerned about code provenance, meaning the chain of title of the code that’s being used by the target. If there are a thousand developers in the world contributing to open source projects, you know, how can the buyer have confidence that each of those developers really had a way to upload particular code. I think that’s a very reasonable concern.

And the problem is not just a lack of regard for intellectual property rights. The problem is that developers don’t necessarily understand the licenses that govern the code they were using. So

they'll combine code, and they'll put it back under terms that a lawyer would recognize as conflicting or incompatible.

But the developer doesn't necessarily. So, buyers are really looking at where the code came from. They want some assurance that the target used a reputable resource.

With respect to the more significant open source organizations out there, those are generally very well run and are very transparent. You can find out who the developers are, for example, for each project. Those organizations also typically have very good legal counsel.

But in addition to those organizations, there are a lot of, you know, individuals out there who just upload code onto the Web. And some of them very clearly do not understand how licensing works. So, buyers want to distinguish those situations. They want to know which resources the target has used so the buyer can decide themselves if they're comfortable with the origins of the code.

So code provenance is one issue. The license terms themselves can also be an issue. Buyers want to know if the target has complied with those licensing terms. Of course, if the target hasn't complied, it may well be breach of contract and copyright infringement. So, buyers want clear information about those potential liabilities and about what it will take to remediate those problems.

Buyers also want to confirm that they're willing to accept the license terms themselves, and that they don't conflict with the buyer's own business model. Now, as – I think we talked a little bit about today – open source licenses really exist on a continuum, from very permissive licenses to licenses that are much more significant and have much more significant terms.

So if I were to take a few minutes to talk about those more significant terms, using as an example the latest version of the General Public License, or the GPL, and why the GPL is not the only license out there with significant terms. It is a hugely popular license, and one you're likely to confront if you handle the kind of due diligence we're covering today.

So to give a little bit of background, as I'm sure many of you know, the GPL was developed by Richard Stallman and his organization called the Free Software Foundation, or the FSF. GPLv2 was developed in the early '90s, and today covers around half of the open source available on the Web. GPLv3, which is the latest version of the license, was published just this past summer. It has some terms that are very similar to GPLv2, but it has also expanded the reach of the GPL into new areas such as patent and digital rights management.

So with that, let me talk a bit about the specific license terms. First, both GPLv2 and GPLv3 are copyleft licenses. Copyleft is a term that was coined by the FSF to describe a philosophy that's the opposite of copyright.

Copyright, of course, is traditionally used to protect works as private property. Copyleft, on the other hand, is the structure the FSF uses to ensure that users have very broad rights to software, including access to source code.

At the heart of copyleft is the reciprocity obligation. Reciprocity means that if you create a work based on GPL software and you distribute that work back out, it has to go out under the GPL. So, in other words, you're required to give the user the source code for the work and those same broad rights to modify and distribute the code that you received under the GPL. Now, that term, alone, conflicts with most commercial software business models. So, if reciprocal code has found its way into a proprietary code base, it will be an issue for the buyer and something they're going to want to fix.

Second, many of the more recent open source licenses have patent provisions. The patent provisions in GPLv3 are particularly significant. First, let me say to the FSF and many other supporters of open source, free patent is a serious threat to open source because patents can be used to interfere with the use and distribution of code.

So, imagine a world where all software were made available under the GPL, and every user was given the same broad rights to copy and modify and redistribute code. Even in that world, if someone holds a patent that reads on the particular code, they can stop others from using and distributing that code. So, that code is no longer free in the sense that the FSF strives for.

So the FSF wanted to find ways to alleviate that threat through GPLv3. And to do that, the FSF added a number of provisions to try to address patents directly. For example, GPLv3 is a very broad patent license. It says that if you make any modifications to a GPLv3 work and then redistribute that work, you automatically give a patent license to all of the patents you have relating to the whole work.

So the patent license isn't limited to the modifications you made. It covers the work as a whole.

Also, the license will automatically extend to any recipient of the work, so it won't just stop with your customer. Under GPLv3, any recipient is entitled to distribute the work, to pass it on. And when they do, it will go with the benefit of that patent license.

There are other patent provisions in GPL as well, covering patent retaliation and some very complicated provisions designed to nullify certain third-party patent license arrangements. I'm not going to go into detail on those today, but suffice it to say that GPLv3 has introduced a whole new level of complexity from a patent perspective. So for those buyers that have (passive) or strategically important patent portfolios, those patent provisions certainly will require the buyer's attention and analysis.

Next, GPLv3 contains an anti-digital rights management provision that's targeted at consumer device manufacturers. The purpose of that provision is simply this; Richard Stallman believes that consumers should have the right to access code on a consumer device. They should be able to modify that code and then reload the modifications on the device. So, to do that, GPLv3 says, when you distribute GPLv3 code on a consumer device, you have to provide source code and all the rights under GPLv3. But in addition, you have to provide all the installation information that a user would need to reload their modified code on the device.

Of course, many device manufacturers are not willing to provide that installation information. They don't want to encourage people to modify the code on their device. So, in their due diligence, device manufacturers are particularly focused on the presence of GPLv3 code.

Lastly, both versions of the GPL and the vast majority of open source licenses contain very broad warranty and liability disclaimers. That's not surprising, given that most of this code is made available for free on the Web.

From a buyer's perspective, that's typically not a deal killer. But in the assessment of the value of the code, if there's no one standing behind it from a legal or operational perspective, you know, that can certainly be an issue.

So those are the legal issues that buyers focus on.

Also, I would also take a minutes and talk about the practical aspects of code analysis. As I was saying earlier, as part of due diligence, buyers today routinely analyze the contents of a target code base. There're technologies available that allow you to automatically detect the presence of open source code in a particular code base. So let's turn to the practical aspects of that kind of analysis.

First, one key issue, of course, in the process is simply the question of who will perform the work, that a buyer can do the work. As you might expect, you know, targets are generally unwilling to agree to that. Targets are worried that the buyer will misuse or misappropriate their technology.

In fact, many buyers themselves don't want to have direct access to the target's code in any event. They know that if the deal doesn't close, the target may make claims that the buyer somehow misused the code. So, neither side generally wants the buyer to handle the work.

We could have the target do the assessment. The target certainly would be happy with that. The buyers, of course, won't agree. They don't believe that the target will really dig in and give them accurate information. So, there is an inherent tension there.

And as you all know, M&A activity is already emotionally charged typically. So because of all of those factors, we must often see that a third party is brought in to do the code analysis. And in that case, the third party really acts as a buffer between the buyer and the target. They're perceived as more objective and less risky in a situation where that often is very much needed.

So, when the buyers decide to bring in a consultant, it's important to think through the logistics for the code analysis. For example, the buyers need to think about where the analysis will take place. Now targets will generally insist that the consultant come to their location. Again, they're not happy to give access to their code to anyone, and if they have to do it, they'd generally want to do it in their own offices. They've a greater sense of control in that situation.

In my experience, targets really run a gamut in their thoughts about the rules of engagement with the consultant. Some are very concerned and very secretive. They have the team show up with nothing in hand, you know, no laptop, no cellphone, no pen and paper. Others are much more open. But in each case, the target needs to think through how it wants to handle the engagement.

And that is particularly true with respect to employees. Depending on when the work takes place, employees may not even be aware that there's an acquisition in the works. So, having a consistent cover story can really be very useful.

I am often asked what I recommend for a cover story. I think the best cover story is intellectual property insurance. So the target will tell its employees that the company is getting IT insurance, and the insurance company has asked for code on it. I recommend that approach because in my experience, the minute you say "insurance," everyone starts listening to you, which, of course, is what you want in a cover story.

But if the target doesn't have a good cover story, then the rumor mill will be very active. Employees are curious, and they'll press for information. And at the very least, it can be a real distraction for the target's business. So, the target's management really needs to be thoughtful about that process.

So, once the report is produced, the buyer will look at what open source the target has in its code base, and they'll think about that from both an operational perspective and a legal perspective. The legal issues are those I talked about. The buyer will look at the origins of the code. Did the target take this code from a hundred different Web sites, or did the target use a couple of reputable sources? The buyer will look at the terms of the license. Are these terms the buyer can comply with, or do they conflict with the buyer's business model? And of course, the buyer will look at the target's compliance of the license terms to see if there're any issues there. Now, the results of that analysis naturally are very different in every deal. I've seen situations where the code that was found really did not present many problems and did not throw the deal down. I've seen other situations where it becomes much more of an issue, and it adds much more risk to the deal, for the target.

So based on the results of the analysis, the target may have to negotiate through remediation plans and potential price reductions and other changes that are generally really not good for the target.

So to summarize, in the M&A context, buyers have become very aware of open source and the risks it can introduce into the target's code base. Buyers want to know what they're buying. They want confidence in what they're buying. So, many major technology players are taking the time to do that detailed code analysis before requiring a particular target. And as that becomes routine for buyers, targets need to understand their code will be analyzed. And the more the target knows in advance, the more prepared they are, the better off they're going to be in that negotiation.

So with that, Maureen, I'll turn it back to you and I'll go back to our first slide for Q&A.

Maureen Dorney: Thanks. So when we first did this Webinar – well, we did this conference in the non-Webinar section of the ACC conference – we actually made it very interactive in a way you can't do it in a Webinar, where we stopped after each section and took questions.

And just to summarize, what I think we saw was every company is dealing with open source now. It's – there almost is no large company or software company of any size that isn't dealing with open source. Everybody's dealing with the same set of problems and issues, and I think the biggest comment that we heard from the group was that it is very, very – it's really (worth) to get the engineers rolling in the same direction in terms of (a sense of) coordinating the engineers and the management so you have a coordinated strategy. And I think what Gemma said at the beginning, emphasizing education, was a reflection of that.

Well, that's it. Let me turn to some of the questions and toss them out for my co-panelists to answer.

First question is, "When is an assignment of copyright needed in connection with contributions to code licensed under an open source type of license? Why not just rely on the terms of the open source license agreement itself instead of asking for an assignment?"

Kat McCabe: Well, I think – this is Kat. The reason you would ask for an assignment is if you want to own the code. I mean that's the – you can certainly rely on typically – depending particularly if you want to use something internally, you can rely on the license coming in. But if your goal, for example, is to have a contractor create something to you that you don't want distributed to other people, then you may want that assignment back into the company.

Maureen Dorney: Have using – have using, Kat or Gemma, of the assignments not just limited to copyright but also covering patents or at least covenants not to assert under patents?

Kat McCabe: I am trying – I don't believe I have, generally, because where I've seen this issue most often is with independent contractors and not typically organizations that have patents.

Maureen Dorney: OK. (The one place) – the one place I'd seen it is at – where the licensee – one of the ((inaudible)) the licensee really wanted to maintain the code base and really wanted to control what was released and when it was released. And also, what one was worried about patents and had a covenant not to assert for patents as well. I don't know, Gemma, if you have anything to add there.

Gemma Dreher: The only thing I was to say is that I think a lot of times, with the assignment of copyright, it's contingent on the governing license. So if we have an engineer that's done something to modify a piece of open source that's governed by the GPL, obviously the GPL is going to continue to govern that modification in the event of distribution. Where if it's governed by

a more liberal license, then it leaves room for the company to decide what it wants to do with that particular modification or derivative work and, you know, how it's going to handle the ownership issues associated with that.

I have not seen many assignments that are broad enough to cover the patents and patent rights as well as the copyright.

Female: OK.

Kat McCabe: Maureen answered the – sorry. I was getting a little feedback there. Moving on to the second question, the question is, “In a consulting project, where the consultants are requested to perform interfacing services between a licensed and an open source software, are the (deliverables) of the interfacing services considered as licensed or open source?” I don't know, Gemma, if you want to take that.

Gemma Dreher: I'm not sure that I'm understanding exactly what this person is getting at.

Female: Well, I think one of the issues there is ((inaudible)) whether the interface that's being developed is going to be – oh – is going to be proprietary or open source, I think.

Gemma Dreher: Well, again, isn't that contingent upon the type of license that is governing the open source piece, as well as the company strategy? Certainly, if the company strategy is to develop things under an open source model – that's why – my confusion arises from some of the variables that we don't know. So let's assume for a moment that the interface is governed by a proprietary license, and it's being integrated or derivative work is being created out of something that's governed by GPLv3, then the derivative work would, in fact, be open source or would be obligated to be open source if it was going to be delivered.

Female: ... just a note on GPLv3. There is permission in GPLv3 to use contractors for work that an organization would do for itself, so it's really just for that organization's internal use. And consultants in that context cannot sign the rights in that code back to the, you know, whoever their customer is. And if the customer continues to use it just internally, there is never an obligation to disclose it to anyone else.

Maureen Dorney: OK, great. And the third question is, “To what extent can a licensee request and successfully obtain third party indemnification provisions in license agreements? For example, to the extent OS is incorporated into licensed software, is it sufficient to receive an indemnification for the licensed software?”

I think, just to (frame) the question before I toss it out, I think this gets back to the situation I mentioned in procurement where if someone – so you are a downstream licensee, there is an open source component, and you want to obtain – you want to obtain third party indemnification provisions for that open source component. Kat?

Kat McCabe: (Yes, I have), you know, in having been in the business of distributing software, I think that to the extent you're distributing open source software almost as a convenience to the customer, you're really passing something through such as the Linux operating system or some other well-known open source project. I think it's fair to say you're not in the business of indemnifying anyone around those existing projects.

But I think, to the extent you're a software developer and you're picking and choosing code and pulling it off the Web, you know, it's certainly not unreasonable for customers to say you're in control of that and we don't – we want you to indemnify us since we have no control on those situations. We think it's only fair that you indemnify us for any problems with that code.

I also think if the code is well integrated with the product, in truth, the software developer is better off having control over those lawsuits because it's something they're going to need to resolve. You know, these things are never – are (really) ((inaudible)) (one-off) situation. If they've got a problem with one customer, they're going to have it with all of their customers.

So (are you going to pass) a number of software developers (trying) to carve out open source and say they weren't going to provide any indemnification? I think that's breaking down a bit in the current market, especially in cases where the developer themselves has decided to tightly integrate that code with their own product.

Maureen Dorney: Yes, I would just add. I think it really has to do with the bargaining power and to the extent that you have a market where you can have a take-it-or-leave-it (advice and send them), and your marketplace needs your solution and you've got (to morally involve) not a lot of negotiating. It works, it can work. To (disclaim) in heavily negotiated licenses, it (very rarely) works.

There is one other very ((inaudible)) that I see which I thought was very interesting recently where a client had incorporated an open source component that was very widely used in the industry had been done under a nonstandard language with some odd wording. But it was generally a very – it was generally a very permissive license that a large upstream OEM licensee – I guess downstream OEM licensee, sorry – was very nervous about the terms because they weren't – didn't fit into one of ((inaudible)) open source licenses, wanted ((inaudible)) to go back and renegotiate with the original open source provider, which had really done it. It really made it (out) and available in free software.

And that proved to be a very difficult and impractical discussion. And I think the idea that you can, without paying a lot of additional money, can go back and change the license terms for open source software that's out (on) the marketplace to get an express indemnity or warranty is very highly unlikely.

We have two questions that (seem to be the same), and I'm going to try to combine them – which is, "(To the extent both involved the) hypothetical that there is software that has – that has incorporated open source software into the OS? And how does that impact ((inaudible)) further redistribution?" – which, I think, is an easy answer, which is a defense on the terms of the license. If it's incorporated into the OS, it depends on how the license that it comes under, and it depends on how the OS is architected. Many people perform – try to (accept) that the open source will be made public (and) try to, under the open source license, (but) try to separate out the – any proprietary program that – any proprietary programs that sit on top of that.

But then the second question, I think, (Kat), is for the GPLv3, which is, "If you were developing a Web site using open source OS(F), what does that mean for the content and software and programs developed on the site that sits on top of that?"

Kat McCabe: (Well), I'm not sure what sits on top of the Web site exactly. There is a movement (afloat) in the GPL context. So GPLv2 was structured so that – and GPLv3 – are structured so that your obligation to provide source code back out under GPLv3 is triggered on distribution of the code.

And I don't know if this is what the question is going after, but I'll talk about it anyway. The – in the context of online providers, to the extent they're using code internally to present a Web site, that's not a distribution, at least under U.S. copyright law. So there is – you know, you find the major online services all use GPL code. They all use Linux, they all use a number of different open source projects.

But they have no obligation to distribute anything back out because they're not a – they're just providing the Web site. That's an internal use. It's generally understood to be an internal use because it's just running on the online service provider's servers.

Now there is a movement (afloat) with something called the Affero General Public License to try to capture modifications made to GPL code where those modifications are provided through functionality in a Web site. So, if you make something available through a network like the Internet, despite the fact that you haven't distributed that work, the Affero license says you still have to provide source code back out under that GPL license. So that's something that would be a radical shift for how this works from an online perspective. It's not something that, at the moment, has much momentum. But I think, as many software vendors move to a software service model, it might be something we'd see becoming a, you know, more controversial and a more – an issue with more momentum in the future.

Maureen Dorney: Thanks, Kat.

Gemma, I'll toss this question to you, if you're OK with that. It says, "One technique we've used to avoid (taking) the proprietary code is to shift the open source component as a separate file. The customer downloads the code directly rather than having it bundled into the proprietary code. Assuming this is in step with a specific license agreement, do you think this is a valid approach?"

Gemma Dreher: It's an approach that I have seen used frequently in my dealings in this area. Kat is certainly much more of an expert on GPLv3 than I am, so I'd be interested to hear what she has to say about it vis-à-vis that particular license.

As to some of the other more restrictive licenses, I haven't seen any challenge to that approach, nor have any of my colleagues that I've discussed with this seen any problems with it.

Kat, as to GPLv3, you ...

Kat McCabe: Yes, I think you're right that the issue becomes more complicated with the GPL because the GPL takes the position that it applies to the entire work, which, if you could see me, I'm putting that in quotes, because the question is, "What does that really mean under the copyright act, and how far does the GPL extend in a software context?"

And there is very little out there. Unfortunately, there's, you know, nothing in the copyright act itself, and there's very little in case law to give us any sense of what constitutes a whole work in the soft – you know, in software, under the copyright act.

So it's – I think, the way the GPL is structured the – what they don't want to see – what the proponents of the GPL don't want to see is people designing around the GPL. So, they don't want a situation where you can break a program into a bunch of different parts and link to those parts, and then say, "Well, but this isn't a whole work that the GPL applies to. It only applies to the specific whole part, and if you download those separately, that's not an issue."

So the question, unfortunately, I don't think, can be answered in a – in the practical sense of how did this code – where did – where did you get the code. I think it's a larger question when you put this code together and it operates together, does that constitute one unified work under the copyright act? And, unfortunately, I think there's – we have very little guidance on what that really – what that really means.

Female: So to manage the issue for a company, if you were to ask your end user if the – are these two pieces of code going to be combined at runtime, so that even if they could be delivered on separate media or segregated out for delivery, and then once delivered, they're combined at runtime, then you might have a problem.

Female: I think that's exactly right.

Female: Yes, I agree with it. That's how I've analyzed the issue.

As we come near the end of the hour, I think one last question. This one just seems to be uniquely for Kat, given her background. The listener asked, “Are there certification providers who perform the audit and certify the report 100 percent open source free or (some) such? And ((inaudible)) what would you get from the report as guidance at the end?”

Kat McCabe: In the M&A context, I don’t think anyone out there certifies that there – it’s (sort of) hard to prove the negative. So, I don’t think anyone would certify that there is no open source code in a particular code base. The point of using the automated solutions is just that you’ll get a lot more information than you could with a manual track because it’s just – that’s just the nature of it. But it’s essentially – I’m going to think what’s being asked, is there insurance that you can get that says, you know, that something doesn’t contain open source code. And I think the answer to that is no, because it’s probably not a – I don’t know if that’s a terrific business to be in, but it may be. We’ll see. We’ll see. But as the use of open source expands, if insurance doesn’t become more widely available – but generally speaking, I think vendors don’t provide that assurance. What they provide is the ability to get at a lot more information than you can in any other way.

Female: Great. Thanks, Kat. One more, just a real quick question. Kat, can you point people to where to find the Affero license?

Kat McCabe: ... Affero is spelled A-f-f-e-r-o, and if you go to the FSF Web site, which I think is fsf.org, you will find the new version of the Affero GPL.

Maureen Dorney: Great. Well, thank you, Kat. Thank you, Gemma.

We are coming to the end of the hour, so I think we’re going to have to stop taking questions for now. I’ve been asked to remind everybody that, again, in your links box is a link to the – to the evaluation. And we’d really appreciate it if you could fill it out.

And with that, our Webcast is now officially concluded. Thank you for your participation.

END